



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

ASSIGNMENT OF BACHELOR'S THESIS

Title: Software module for skill based search of students
Student: Adam Jankovec
Supervisor: doc. Ing. Pavel Kordík, Ph.D.
Study Programme: Informatics
Study Branch: Web and Software Engineering
Department: Department of Software Engineering
Validity: Until the end of summer semester 2019/20

Instructions

Explore methodologies of evaluating and searching for students based on their study results.
Analyze the Grades application used at FIT CTU.
Collect requirements, design, implement and test a system enabling a skill-based search of students based on their results in Grades.
Apply methods of Software Engineering during the design and implementation of the system providing features for manual defining of own skill set, evaluation of skill sets and adding references.
Test functionality and usability and discuss ways of extending the system.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 6, 2019



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

Bachelor's thesis

System for a skill-based search of students

Adam Jankovec

Department of Software Engineering
Supervisor: doc. Ing. Pavel Kordík, Ph.D.

May 6, 2019

Acknowledgements

First of all, I would like to thank my supervisor doc. Ing. Pavel Kordík, Ph.D., and Ing. Stanislav Kuznetsov for investing their time into our sessions. Many thanks to Ing. Eliška Šestáková and everyone from 341b, namely: Ing. Václav Blažej, Ing. Štěpán Plachý, Bc. Tung Anh Vu, and Tomáš Vopat, for their reviews, advice, and support. Thanks to everyone who gave valuable feedback during the process of collecting requirements and also to testers, whose effort will result in improvement of the system. Last but not least, thanks to my family and my girlfriend for their support.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on May 6, 2019

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2019 Adam Jankovec. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Jankovec, Adam. *System for a skill-based search of students*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Tato práce se zaměřuje na vytvoření webové aplikace pro vyhledávání studentů podle dovedností, která generuje studentům dovednosti ze známek, sesbíraných během studia na Fakultě informačních technologií Českého vysokého učení technického v Praze. Výsledkem práce je koncept, přinášející možnost filtrovat studenty podle jejich množiny dovedností, který mohou firmy a učitelé využít při vyhledávání kandidátů pro svá zadání.

Klíčová slova webová aplikace, vyhledávání podle dovedností, dovednost, známka

Abstract

This thesis focuses on the development process of a web application for a skill-based search of students, which generates student skill sets from grades, collected during students' study at Faculty of Information Technology at Czech Technical University in Prague. The result is a proof of concept, which brings a possibility to filter students based on their skill set, which can be used by companies and teachers to search for candidates for their assignments.

Keywords web application, skill-based search, skill, grade

Contents

Citation of this thesis	vi
Introduction	1
1 Background & State of the art	3
1.1 Skill mining	3
1.2 Grades portal	4
1.3 Related existing solutions	5
1.3.1 LinkedIn	5
1.3.2 Candidate Search	6
1.3.3 Cooperation with Industry portal	7
1.3.4 Summary	8
2 Preliminaries	9
2.1 Technology stack	9
2.2 Database design process	13
2.3 Choice of server-side architecture	14
2.4 Building a web application with Angular	14
2.4.1 Material Design	15
3 Analysis	17
3.1 Requirements engineering	17
3.1.1 Concept	18
3.1.1.1 Profile page	18
3.1.1.2 Search page	20
3.1.2 Collecting feedback and requirements	21
3.1.2.1 Companies	21
3.1.2.2 Teachers	22
3.1.2.3 Students	24
3.1.3 Summary of requirements	24
3.1.3.1 Functional requirements	24

3.1.3.2	Nonfunctional requirements	26
3.2	Use case modelling	26
3.2.1	Actors	26
3.2.2	Main use cases	26
4	Architecture	31
4.1	Client-server architecture	31
4.2	Server-side architecture	32
4.2.1	Architecture description	32
4.2.2	Code structure	34
4.3	Client-side structure	36
5	Design	37
5.1	Profile	37
5.1.1	Sections	39
5.2	Skill points calculation	43
5.3	Searching for profiles	44
5.3.1	Search by name	45
5.3.2	Advanced search	45
5.4	Database model	47
6	Implementation	49
6.1	Requirements	49
6.2	Server-side development	49
6.2.1	Getting data from Grades	50
6.3	REST API	50
6.4	Security	51
6.4.1	Authorization	51
6.4.2	User roles	51
6.5	Client-side development	52
6.5.1	Resolving CORS policy violation	53
7	Testing	55
7.1	Automated testing	55
7.2	Static code analysis	57
7.2.1	SonarQube	57
7.3	Usability testing	58
7.3.1	Test script for companies	59
7.3.2	Test script for teachers	59
7.3.3	Test script for students	61
7.3.4	Changes proposed by participants	62
7.4	Acceptance testing	64
8	Ideas for extensions	67

Conclusion	69
Bibliography	71
A Acronyms	75
B Screenshots of the result	77
C REST API definitions	85
D Contents of enclosed DVD drive	89

List of Figures

1.1	Process of skill mining	4
1.2	Example of user’s skill activity on Stack Overflow	6
1.3	Example of an assignment published in SSP	7
2.1	Material Design example	15
3.1	Profile page concept	19
3.2	Search page concept	20
3.3	Use case diagram	27
4.1	Deployment diagram	31
4.2	Layered architecture	33
4.3	Module dependencies	35
5.1	Profile prototype from teacher’s perspective	38
5.2	Detail of Introduction section	39
5.3	Detail of Study Information section	39
5.4	Detail of Study results section	39
5.5	Detail of Interests section	40
5.6	Detail of Status section	40
5.7	Detail of Contacts section	40
5.8	Detail of Projects section	41
5.9	Detail of Professional experience section	41
5.10	Detail of References section	41
5.11	Detail of Subjective proficiency section	42
5.12	Detail of Skill cloud section	43
5.13	Sequence diagram of processing grades into skill points	44
5.14	Search page prototype	45
5.15	Database model	48
6.1	Authorization flow	52

6.2	Proxy configuration	53
6.3	Request translation via Proxy	53
7.1	Example of testing database transaction with DbUnit	56
7.2	Example of dataset consumed by DbUnit	56
7.3	Example of mocking an external data source	56
7.4	Final report generated by SonarQube	58
7.5	Graphs of participants' opinions about the user interface	64
7.6	Profile page	66
B.1	Login page	77
B.2	Authorization page	78
B.3	Profile page	79
B.4	Edit subjective proficiency page	80
B.5	Edit reference dialog detail	81
B.6	Search bar with autocomplete detail	81
B.7	Search page without parameters	82
B.8	Skill cloud for browsing most used skills	82
B.9	Search page with parameters and results	83
B.10	Settings page	83

List of Tables

3.1	Functional requirements realisation	30
6.1	REST endpoints implemented in Grades	50
7.1	Test coverage of the Data Layer	57
7.2	Task completion of companies	59
7.3	Task completion of teachers	60
7.4	Task completion of students	62
7.5	Changes proposed by participants	62

Introduction

The task of searching for relevant candidates for jobs in the field of information technology is becoming more difficult with the constantly growing demand. With the lack of people in the IT (Information Technology) industry, many companies are reaching out to students in their early years of college, wanting to raise their future employees, to ensure a steady flow of fresh graduates. For students of FIT (Faculty of Information Technology) CTU (Czech Technical University in Prague), who are not looking for employment just yet, there are contracts for one-time work offered by the companies. In addition to companies, teachers are also publishing interesting assignments for school projects or theses. Students gain experience and a corresponding amount of money as a reward for working out these contracts, but the majority of these assignments will go unnoticed, and students are missing an opportunity to gain experience while working on exciting projects.

Both the teachers and the companies are searching for one thing – students with a particular skill set, who are willing to collaborate. Instead of letting students browse hundreds of assignments aimlessly, the approach could be reversed by recommending the assignments to students according to their profiles. Student profiles would comprise of preferences and student's own skill set, which would be generated using educational data (grades), related to the student's study. The profiles would be used to search for candidates, who are relevant for published assignments. The system could lead to an increase in work experience of students and has the potential for attracting new faculty partners and change the way faculties and companies cooperate.

The main aim of the thesis is to create a system that allows a skill-based search of students. The system should be connected to the Grades application, running at FIT CTU, to process student's grades and generate skill points from them. Among the goals are also the following: collecting requirements on the system, designing the system and implementing its core parts, testing the functionality and usability of the solution appropriately, suggesting possible ways of extending the system.

This thesis consists of eight chapters. The first chapter describes the background to skills and analyses related state-of-the-art solutions. The second chapter (Preliminaries) explains used technologies, introduces requirements for database design, selects appropriate server-side architecture, and describes modern client-side development. The third chapter (Analysis) concerns the process of gathering requirements, creating a concept of the system and use-case modelling. The fourth chapter (Architecture) focuses on the realisation of non-functional requirements and describes the structure of the client and the server. The fifth chapter (Design) is about the realisation of functional requirements, as well as, creating and explaining a prototype of the system. The sixth chapter (Implementation) describes the steps taken to overcome obstacles, which were encountered during development. The seventh chapter (Testing) focuses on automated testing, usability testing, quality assurance, and acceptance testing. The eighth chapter (Ideas for extensions) discusses possible ways of extending and improving the system.

Background & State of the art

This chapter describes the process of skill mining and its use in the affiliated systems (Grades, Cooperation with Industry). It also dedicates a section to the analysis of the state-of-the-art solutions for a skill-based search of people.

1.1 Skill mining

“Educational data mining is an emerging discipline, concerned with developing methods for exploring the unique and increasingly large-scale data that come from educational settings and using those methods to better understand students, and the settings which they learn in.” [1]

The process of converting educational data (grades) into skills is a problem, which has already been addressed by Ing. Stanislav Kuznetsov in his doctoral study [2], where the author mentions, that *“Universities cannot use their educational data efficiently because the data are often not utilized due to confidentiality and security reasons.”* By transforming the confidential data (grades, courses) to a non-confidential format (skills), the data can be utilized to generate profiles of students and lecturers. [2, p. 14]

Skill mining, as shown in Figure 1.1, consists of *“extracting accreditation materials from a data warehouse, these materials are then automatically classified into a set of keywords, acronyms, and text. Text mining is performed next, thanks to which we will obtain ‘proposed skills’. Additionally, we created a website to collect information from teachers, where each teacher shall indicate what skills belong to a subject based on the ‘proposed skills’”*. [2, p. 14–15]

The skill mining algorithm brought mixed results. It managed to catch many useful skills, but the result still contains a significant amount of noise. Generated skills were inserted into the database of the Grades portal [3] with a resulting number of approximately 50 000 skills. These “proposed” skills will be used for debugging during development and for a final presentation of the thesis, but they should be replaced with a more accurate set before deployment of the system.

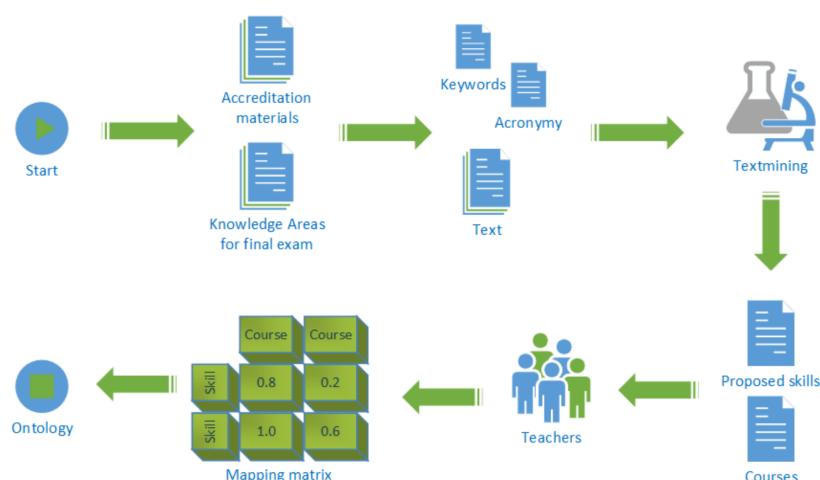


Figure 1.1: Process of skill mining. Source: [4]

1.2 Grades portal

Grades [3] (also known as Klasifikace) is a portal for managing courses and evaluating students. The portal is currently used at FIT CTU as one of the two main evaluation systems and has an increasing number of supporters among teachers as well as students. Ing. Zdeněk Balák developed a prototype of the system as a part of his master thesis during the summer semester of 2017/2018 [5]. From then the Evolution team¹ continued the development of the system by contributing with several major releases, each refining and tweaking the system to the users' needs. The last major release was support of external applications, which allows other school systems such as ProgTest² [6] to write points to students' evaluations via secured REST (Representational State Transfer) endpoints.

The core functionality of the system is creating course definitions (tests, exams, homework, ...) evaluating students using several evaluation views with "course evaluation", "detailed course evaluation", "student evaluation". To improve teachers' comfort, Grades offers features such as expressions, which automatically calculate definition values.¹ Users can submit issues and suggestions via GitLab Issues³, and by having a light-weight, faculty-specific solution, Evolution team can tailor the system to match users' needs.

¹A team of developers from 341b at FIT CTU, currently consisting of the members: Ing. Václav Blažej, Adam Jankovec, Ing. Štěpán Plachý, Tomáš Vopat, Bc. Tung Anh Vu.

²A task evaluation system, used at FIT CTU.

³<https://gitlab.fit.cvut.cz/evolution-team/classification-issues/issues/>

Grades already contains basic support for adding skills to courses. Course editors can create new skills and assign existing skills to courses with a certain weight ranging from 1 to 100 percent. The selected weight is directly proportional to the importance of skill in course. For a more detailed distribution of skills, editors can also assign skills to course's definitions.

1.3 Related existing solutions

The idea of searching for people by their skills already has dozens of available solutions. This section examines the most relevant sites for searching job candidates and analyses their strengths and weaknesses.

1.3.1 LinkedIn

LinkedIn [7] is the biggest social network for professionals in the world, with over 546 million members from 200 countries. It was founded in 2002 and as a result of LinkedIn's success, Microsoft finished its acquisition in December 2016. [8] It is a place for anybody who is interested in seeking a job or is recruiting someone else for a job.

Users can create a profile and specify many important details about themselves. They can set a profile photo, motto, education, work experience, skills, certificates and languages. Users can also add other users into their network, and these connections can endorse their skills and give them references. Users can message each other and add posts on their "wall", which can be rated and responded to by other users.

The filter has two options. The first option is to search for *people* and the basic version provides the following parameters: connections level, connections of a particular user, location, current companies, past companies, industries, profile language, and schools. The second option is to search for *jobs* with these parameters: date posted, job type, company, industry, job function, experience level, title. These filters are missing a vital parameter – skills. LinkedIn profits on its membership fees. Basic accounts have all the functionalities required to create a profile and connect with people. On the other hand, premium recruiters have access to more detailed filtering with extra parameters such as the mentioned *skills*, which is a smart move from LinkedIn because it is a powerful feature recruiters are willing to pay for.

One of the strongest points of LinkedIn is that its fanbase is enormous, so the chances of finding relevant candidates are quite high. However, the main issue of LinkedIn is that with its growing number of members, a lot of spam began to appear. Recruiters sometimes just copy and paste their proposals without even checking if the profile is relevant for the job. Another problem is that a significant number of accounts is inactive or not looking for employment. This can be a major turnoff for both the recruiters and the candidates, who waste their effort while having to endure these problems.

Another issue is related to skill endorsements, where anyone can endorse other people's skills without any guarantee. This degrades the endorsements' value and recruiters may stop paying attention to it. This fact will be taken into account, and only teachers will be able to endorse students' skills.

1.3.2 Candidate Search

Candidate Search [9] is a solution developed by Stack Overflow to help in search for developer candidates. It is built around an idea of resume being a thing of a past and offers a way of analysing Stack Overflow profiles, which can tell much more than just a simple resume.

“Stack Overflow is visited by over 50 million people every month – and is the largest, most trusted online developer community for developers to learn, share their knowledge, and build their careers. Professional and aspiring programmers visit Stack Overflow each month to help solve coding problems, develop new skills, and find job opportunities.” [10]

Users can not define their skills, but they have an option to set job preferences through “techs they would like to work” with, “techs they would prefer not to work with”, roles with experience level and job type. To replace subjectively defined skills, user's knowledge is measured in activity with the skill. All questions and answers related to a certain topic can receive upvotes, downvotes, and even badges. User's understanding of a topic can then be measured using the number of points earned on related posts, as depicted in Figure 1.2. Rating user's knowledge of the topic by the amount of activity is a brilliant approach, the only problem could arise if the user decides to pursue a different path and would like to distance himself from the previous skills. This way he would still get connected to the old field of IT.

Candidate Search is a great solution and offers lots of things for inspiration. The fact that it is built solely for the purpose of aiming at IT industry candidates, gives it a significant advantage over more generalized solutions.

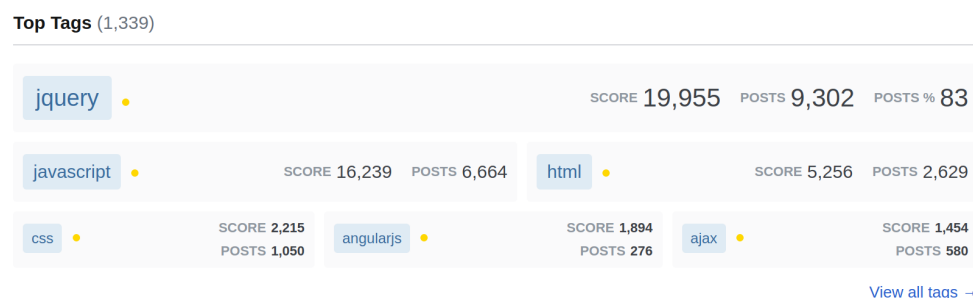


Figure 1.2: Example of user's skill activity on Stack Overflow. [screenshot was captured by the author]

Better targeting of win-back campaigns ✓ APPLY Show details →

machine learning Machine learning clustering

We have sent out 1.2 million vouchers to some of our customers. A portion of the vouchers have not been used. We would like to investigate what are the similarities between the customers who used the offer and those who didn't.

showmax

3.11.2017 - 31.12.2020 60 hours 20000 CZK / team 333 CZK / hr.

Positions:

1x [Data Engineer](#)

Figure 1.3: Example of an assignment published in SSP. [screenshot was captured by the author]

1.3.3 Cooperation with Industry portal

Cooperation with Industry [11] (also known as SSP or Spolupráce s Průmyslem) is a portal for connecting industry partners, teachers and students.

Industry partners can publish assignments as shown in Figure 1.3. These assignments contain a detailed description, estimated number of hours to complete, amount of money as a reward and a required set of skills. Teachers validate the assignment and students can then apply for it either as individuals or as a team. Students can also gain points in school, by submitting their solution as a semestral project in related courses.

SSP is one of the older projects developed by the Evolution team, but unlike Grades, it is built on a LifeRay DXP (Digital Experience Platform) [12] technology. Liferay is a platform for building portals, intranets and websites, which are both scalable and responsive. Although LifeRay provides fine, pre-written solutions, the Evolution team considers SSP to be an old, monolithic project, written with old coding habits and will probably be replaced in the future by a more light-weight solution.

SSP was the first to introduce processing of grades into skills, but the process was never automated. Data had to be manually exported out of KOS (KOMPONENTA Student)⁴ [13] and imported into SSP for processing (discussed in Section 1.1) and generating skills of students. There is also no possibility of searching for students by the requested skills, which is why a skill-based search is a topic this thesis focuses on. Although the search could be implemented

⁴Study information system used by CTU.

into SSP, it was decided that the existing code is not going to be extended with this new feature because the old code no longer complies with the current standards of the Evolution team. Therefore, a new solution is going to be built using the latest technologies.

1.3.4 Summary

Solutions, mentioned in the previous sections, proved to be inappropriate to be used for searching among students. It is critical to be able to restrict people's access to students accounts via roles, which none of the solutions provide. It is also required to be able to restrict giving skill endorsements only on the teacher role, which would secure the value of endorsements and not degrade it. Another missing requirement is to set the size weight of skills. All of them were either same value or divided into "top" and "other", which is not sufficient. The last problem is the inability to manually set a skill set generated from grades of students.

The reasons mentioned above lead to the decision of having a custom made light-weight system, which would allow the faculty to have its network of profiles. All mentioned downsides will be taken into account during design to make sure to learn from their weaknesses. Their strengths, however, will serve as an inspiration for improving the new solution.

Preliminaries

This chapter focuses on the technology stack used by Grades and discusses the differences in versions, of both Grades and the new system. It also describes the database design process, choice of server architecture and development of application with Angular.

2.1 Technology stack

This section describes the technology stack used in Grades. Since the Evolution team will continue in development of both Grades and Skills, both systems should be built on similar technology stacks for easy maintenance, but versions may vary.

Java – Programming language suitable for larger enterprise projects. At the time of implementation, Java 11 was the latest version. Java 12 was released in March 2019 [14], and migrating to it would require only a small effort, but Java 11 remains the main choice because it offers long term support [15].

Maven – Tool for building and managing Java-based projects. Based on the concept of a project object model (POM), Maven can manage a project’s build, reporting and documentation and dependencies. Project dependencies can be maintained in the module’s POM file, where the developer has to specify groupId, artifactId, and version to identify the requested dependency.

ORM – Object-Relational Mapping is used when there is an effort to map objects on database tables. In the application, real-world objects are represented as classes, and in a relational database, an entity class is represented as a table, with rows representing a class’s instance. The main goal of ORM is synchronization between the objects, used in the

application, and their representation in the database system to ensure data persistence.

JPA – “*The Java Persistence API is the Java API for the management of persistence and object/relational mapping in Java EE and Java SE environments.*” [16] JPA is merely a specification. It is a set of rules, which are to be followed when implementing a solution.

An entity is a persistence domain object, which represents an entity table in a relational database and each instance corresponds to a row in a table. According to [17] “*an entity class must follow these requirements:*”

- *The class must be annotated with the `javax.persistence.Entity`.*
- *The class must have a public or protected, no-argument constructor. The class may have other constructors.*
- *The class must not be declared final. No methods or persistent instance variables must be declared final.*
- *If an entity instance be passed by value as a detached object, such as through a session bean’s remote business interface, the class must implement the `Serializable` interface.*
- *Entities may extend both entity and non-entity classes, and non-entity classes may extend entity classes.*
- *Persistent instance variables must be declared private, protected, or package-private, and can only be accessed directly by the entity class’s methods.*”

Entity relations can have multiplicities, and these multiplicities can be either unidirectional (only one has reference to the other) or bidirectional (both have reference to the other). There are four types of multiplicities:

One-to-one – Instance of one class has reference to an instance of another class.

One-to-many – Instance of one class has references to many instances of another class.

Many-to-one – Many instances of a class have reference to one instance of another class.

Many-to-many – Many instances of one class have references to many instances of another class.

Hibernate – Hibernate is an implementation of JPA. Mapping of classes to database tables (ORM) can be done using either XML (Extensible Markup Language) or Java Annotations. “*Hibernate uses a powerful HQL (Hibernate Query Language) that is similar in appearance to SQL (Structured Query Language). Compared with SQL, however, HQL*”

is fully object-oriented and understands notions like inheritance, polymorphism and association.” [18] Hibernate is also database independent and handles the adaption to a new database. Therefore, the developer can use the same code across multiple databases.

Hibernate also provides an entity validator, which allows to express and validate application constrains. Restrictions can be applied using either XML configuration, or Java Annotations (e.g., `@NotNull`, `@Max(255)`).

PostgreSQL – Open source object-relational database system that uses and extends the SQL language. According to developer survey carried out in 2019 [19], PostgreSQL turned out to be the second most used relational database system among the developers.

PostgreSQL databases can be managed via pgAdmin which is a database administration and development platform. It provides database operations such as “create”, “backup”, and “restore” from backup.

Spring Framework – Open source framework, which provides a comprehensive programming and configuration model for modern Java-based enterprise applications. Spring’s infrastructural support allows the developers to focus on the application-level business logic. [20]

Spring is also labelled as an IoC (Inversion of Control) container. This means that it adopts a programming principle called Inversion of Control and is, therefore, responsible for managing object lifecycles: creating these objects, calling their initialization methods, and configuring these objects by wiring them together via dependency injection.

Spring is modular by design. It consists of about 20 modules, which can be easily imported into any Java project. *“These modules are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, Messaging, and Test.”* [21]

Spring Boot – An extension of the Spring Framework, which helps eliminate boilerplate configurations to speed up the development process. It allows the developer to focus on developing business logic, rather than wasting time with project configuration.

Spring Boot is modularized, and required modules can be added into a project as Maven dependencies by importing “spring-boot-starter” modules, which contain all that is required for certain functionality, for example, connecting a database or setting up REST controllers.

REST – REpresentational State Transfer, is an architectural style for providing standards between computer systems on the web, and making it easier for systems to communicate with each other using familiar constructs such as HTTP (Hypertext Transfer Protocol) status codes.

RESTful systems are distinguished by a separate client and server. [22] The representation of the state can be serialized into many formats such as XML, HTML, YAML, but JSON (JavaScript Object Notation) is currently the most widely adopted format.

REST is created as an HTTP abstraction and adopts HTTP principles, but abstracts its technical details. It is the leading programming model for creating Web APIs. In addition, a system, which complies with the principles of REST is called RESTful, and to manipulate resources, it uses the following set of HTTP methods:

GET – Request a resource without modifying it.

POST – Create a resource.

PUT – Create or edit a resource. This method is idempotent⁵ and previous resource is always overwritten by the new one.

DELETE – Delete a resource.

PATCH – Edit a resource. Only the fields which are in the new object will be overwritten by the old ones.

OAuth 2.0 – *“OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 supersedes the work done on the original OAuth protocol created in 2006. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices.”* [23]

“The Authorization Code grant type is used by confidential and public clients to exchange an authorization code for an access token. After the user returns to the client via the redirect URL (Uniform Resource Locator), the application will get the authorization code from the URL and use it to request an access token.” [24]

TypeScript – It is a *“typed superset of Javascript that compiles to plain Javascript”*. [25] One of the main benefits of static typing is that IDEs (Integrated Development Environment) can spot common errors during coding, which results in a more efficient development. For a large JavaScript project, adopting TypeScript might result in more robust software, while still being deployable where a regular JavaScript application would run.

Angular – Open source web application framework, which was released by Google and is also known as “Angular 2+”. Angular provides a command line interface called Angular CLI, which can be used to create projects, generate components, and deal with testing and deployment.

⁵It can be called repeatedly and will always return same response.

Angular’s component based-architecture promotes high quality, encapsulated, reusable code, which is also unit-test friendly and maintainable, due to easy decoupling and replacing of components. It also utilizes hierarchical dependency injection to improve performance.

Although Angular is based on TypeScript, AngularJS (a predecessor of Angular) is based on JavaScript. Grades client is written in AngularJS, and since AngularJS and Angular are vastly different, a goal was set to have versions of both Grades and Skills systems up-to-date. Therefore, at the time of writing this thesis, a complete rewrite of the Grades client into the newest Angular 7 is underway.

2.2 Database design process

“The relational data model, which organizes data in tables of rows and columns, predominates in database management tools. Today there are other data models, including NoSQL and NewSQL, but RDBMSs (Relational database management system) remain dominant for storing and managing data.” [26] Data model needs to be relational due to the fact, that PostgreSQL is a relational database and is a part of the technology stack.

According to [27], *“Designing an efficient, useful database is a matter of following the proper process, including these phases: requirements analysis, organizing data into tables, specifying primary keys and analyzing relationships, normalizing to standardize the tables.”*

“Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals, as they reduce the amount of space a database consumes and ensure that data is logically stored.” [28]

“The database community has developed a series of guidelines for ensuring that databases are normalized. These are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF).” In most designs, there are only three normal forms. [28] According to [29, slide 156, translated by the author], the first three forms can be described as follows:

1NF – Attributes are atomic (no structured, multi-value attributes).

2NF – All non-key attributes depend on the whole key (no partial dependency on the key).

3NF – All non-key attributes depend on the key directly (no transitive dependency on the key).

2.3 Choice of server-side architecture

During the process of deciding which architecture is suitable for the Server, the emphasis was put on ease of development and good testability. The analysis conducted in [30, Appendix A] shows, that Layered architecture (also known as N-tier) is the most appropriate for such case, although it has its downsides, such as lower performance and scalability.

“Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic). Although the layered architecture pattern does not specify the number and types of layers that must exist in the pattern, most layered architectures consist of four standard layers: presentation, business, persistence, and database.” [30, Chapter 1]

“Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request. Each layer in the architecture forms an abstraction around the work that needs to be done to satisfy a particular business request. For example, the presentation layer doesn’t need to know or worry about how to get customer data; it only needs to display that information on a screen in particular format. Similarly, the business layer doesn’t need to be concerned about how to format customer data for display on a screen or even where the customer data is coming from; it only needs to get the data from the persistence layer, perform business logic against the data (e.g., calculate values or aggregate data), and pass that information up to the presentation layer.” [30, Chapter 1]

2.4 Building a web application with Angular

“The basic building blocks of an Angular application are NgModules, which provide a compilation context for components. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules. An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.” [31]

“Components define views, which are sets of screen elements that Angular can choose among and modify according to program logic and data, while also utilizing services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making code modular, reusable, and efficient.” [31]

To make sure not to reinvent the wheel again, there are projects, which provide sets of already written components for developers to reuse. Angular Material is an open-source project for developing a range of components, which

implement common interaction patterns according to the Material Design specification. The project provides a set of reusable, tested components, with support of accessibility.

2.4.1 Material Design

“Material is an adaptable system of guidelines, components, and tools that support the best practices of user interface design. Backed by open-source code, Material streamlines collaboration between designers and developers, and helps teams quickly build beautiful products.” [32]

“Material Design emerged as Google’s brainchild in mid-2014 (...) the goal is to deliver high-quality output consistently across platforms, giving users control over clearly indicated, pleasant-looking components that behave like real-world objects. Material Design involves applying basic, natural laws from the physical world, principally concerning lighting and motion.” [33]

“The idea is that by mimicking the physical world, users’ cognitive loads are reduced through careful attention to layout, visual language and pattern library, maximizing predictability and eliminating ambiguity.” [33]

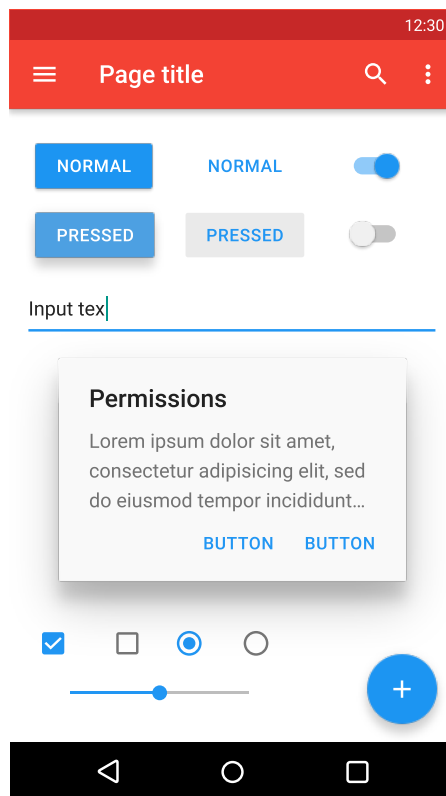


Figure 2.1: Android application showing interactive elements according to Material Design. Source: [34]

Analysis

This chapter focuses in detail on the process of collecting requirements. First, a set of requirements is collected from the supervisor's idea of the system. The initial concept is designed and then taken to collect more requirements from the interested parties. A thoroughly described summary of requirements is followed by use case modelling.

3.1 Requirements engineering

It took several sessions to capture supervisor' notion of the system. The idea is as follows. Students are going to have their own profiles, which will contain references received from teachers, their own subjectively defined skill sets, and generated skill sets. The system will be collecting educational data from Grades, in the form of students' grades. These grades will be used to generate skill sets, which represent knowledge student should be familiar with. Finally, the system will provide an option to search for students based on their skill set, which will be available to students, teachers, and external partners of FIT CTU (now referred to as companies). Initial set of requirements, gathered from the sessions, was documented as follows:

- parameterized search of students,
- skill endorsement,
- references,
- subjective skills,
- generated skill sets.

3.1.1 Concept

The plan was to create a concept, which could then be presented to all the interested parties. The created concept consists only of a simple profile page and search page. It does not depict the website's layout.

3.1.1.1 Profile page

The concept of the profile page, illustrated in Figure 3.1, includes all the required features and also some experimental features, which were added as an attempt to make the profile more interesting. The profile is divided into these sections:

Chart of interests – Should reflect the student's field of interest. It can be either student's level of knowledge, or level of interest. Basic idea is to split the chart into several sections, each representing one field of IT, and to let the students choose the levels themselves.

Personal information – General information about student's field of study, year and programme.

Contacts and links – Contact information and links to student's projects on sites such as GitLab or GitHub.

Skills match – Detailed description of matched and unmatched skills based on search parameters. Match is represented in percentages.

References – Personal references received from teachers, which were given for collaboration on projects, extra activity, above-average results, etc.

Endorsed skills – Teachers have an option of endorsing student's skills. Endorsements can also contain optional comment, about the reason behind endorsement. Endorsement should be only given to those who proved their knowledge of the topic, or the best ones of class and not to everyone who passes some test.

Skill chart – Multilevel pie chart represents generated skill set. Inner circles are higher level skills (more abstract). Skills become more specific with each outer level. Levels contain bigger and smaller nodes. The bigger the node, the more a student is familiar with the topic. Each parent node's size is a sum of children values.

3. ANALYSIS

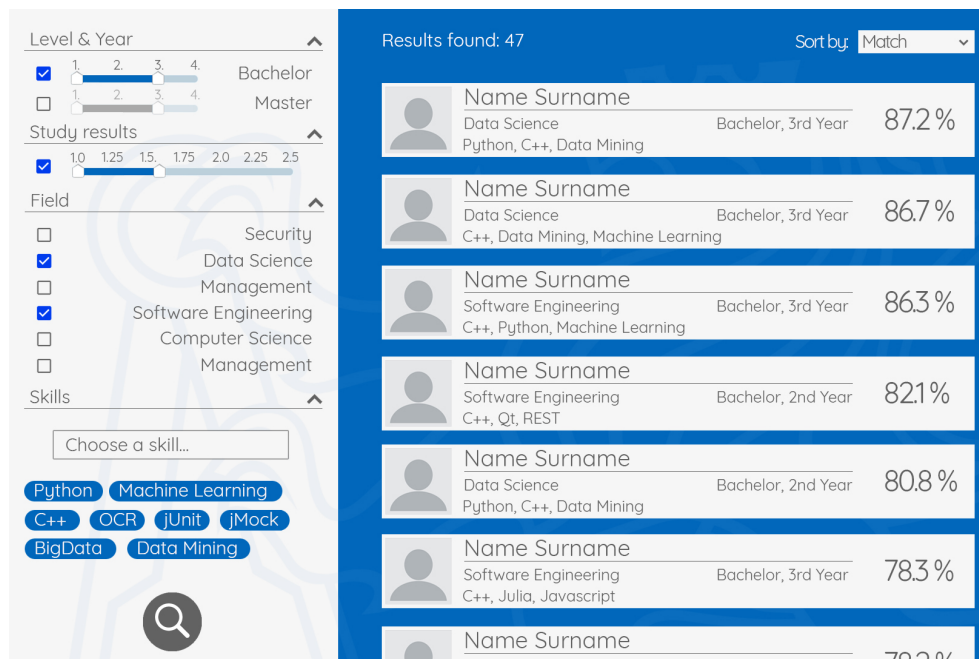


Figure 3.2: Search page concept

3.1.1.2 Search page

Concept of the search page shown in Figure 3.2 is divided into filter section and results section. Users have an option to filter by the following parameters:

Level & Year – Level and year of study,

Study Results – Range of weighted study mean,

Field – Field of study (e.g., Software Engineering, Hardware, ...),

Skills – Set of skills.

After the filtering is done, a list of matched profile previews appears and each preview contains:

Study information – Field of study,

Skills – Most rated skills by the amount of skill points and endorsements,

Match – Percentual match of skills.

3.1.2 Collecting feedback and requirements

Due to the fact that there are three groups of users (students, teachers, companies), the process of collecting requirements is divided into three parts. Each part describes the suggestions given by the group's representatives.

Since the process of introducing the participants to the system could be a complicated matter, a qualitative approach was chosen instead of the quantitative. By selecting the qualitative approach, it was possible to have a more in-depth conversation about the system, and all the questions, participants had, were answered immediately. A substantial amount of time was invested as some conversations were lasting even longer than an hour. However, the participants' ideas lead to a considerable improvement of the concept.

Participants were introduced to the issue of not being able to filter students by some criteria. Then a possible solution was presented using printed copies of the profile and the search page concepts. Contents of the search page, illustrated in Figure 3.2, were described first, followed by the search results and a detailed description of the profile page and its sections, depicted in Figure 3.1. After presenting the concept, participants were asked the following questions:

- “Are there any functionalities you would like to add?”,
- “Are there any functionalities you don't find relevant?”,
- “Would you use the system, if there was the option?”.

3.1.2.1 Companies

Collecting requirements from companies occurred during the faculty career fair COFit, which took place on 24. October 2018. Since lots of potential users were present on that day, it was an ideal opportunity to receive feedback. Among the companies were: Greyson, Attacama, eMan, Etnetera, Gemalto, OKSystem, Quanti, Datamole. Companies are listed only to illustrate the participant pool size and their names will not be listed in the answers. Out of nine companies, eight were impressed with the idea, and six would probably use the system.

Functionalities to add

- Availability status

Description: “Filtering profiles is pointless if the student has no interest in collaboration. Student should have an option to specify types of collaboration such as full-time or part-time.”

Response: Section “Availability status” indicating students' interests will be added.

3. ANALYSIS

- Professional experience

Description: “Student’s knowledge would gain more weight, if there was a way to see what he had learned from practice. Student should have an option to add his professional experience with positions and specify the job description and projects he participated on.”

Response: Section “Professional experience” will be added as an option for students to specify their job positions and their description.

- Searching for senior position candidates

Description: “The system should be able to search people for senior positions in some way. Alternatively, stick to the profiles of people who have already graduated.”

Response: Interest in senior positions will be recorded as a low priority feature request and will not be implemented in the proof of concept.

Irrelevant functionalities

- References

Description: “It is not essential for us, that someone from the school gave the student a positive comment. We do not know the people who grant them, and we will check the students ourselves.”

Response: Since only one company has identified the references as irrelevant, they will not be removed from the system.

3.1.2.2 Teachers

Teachers were approached during October and November 2018. Department heads were approached first, followed by several teachers belonging under the departments. In addition to collecting requirements, the goal was to expand the awareness of the system. Furthermore, it was necessary to ensure the timely allocation of skills to subjects in Grades, from which points could be generated for students. Reactions on the system were mostly positive.

Functionalities to add

- Availability status

Description: “A student should have an option for binary switching between I want / don’t want to be reached out to.”

Response: The availability status has already been mentioned in the suggestions of the companies. Therefore, it is resolved.

- Subcategories of study fields

Description: “It would be more convenient to be able to filter by some subcategories of fields, which would more closely specify focus if we are not looking for specific technologies.”

Response: Filtering by study field subcategories will be recorded as a low priority feature request and will not be implemented in the proof of concept.

- Higher skill points reward for bachelor/master thesis

Description: “Students working on a bachelor/master thesis in specific technologies usually leave with more knowledge than after completing a course with a given technology. They should, therefore, receive a corresponding evaluation.”

Response: The work has a corresponding higher number of credits. Thus, the number of credits will increase the amount of skill points generated for the thesis.

- Ability to browse student profiles

Description: “Students should also be available for the overview itself. They should be visible even if they are not interested in work.”

Response: The system will allow setting profile visibility. Search bar for searching by name will be added.

- Modify the percentage matching system

Description: “Allow adding weight to searched skills, for example, search by a set of skills with 1/2 C++, 1/4 Java and 1/4 Javascript. This would allow to emphasize which skills are more important. Students should also have an option to specify what skills are important to them, to filter out those with active interest in the topic.

Response: Adding weight to searched skills would add more complexity to filtering and could discourage new users. The search without weight will be retained. The student’s own interest by the interest in the study field and subjective proficiency.

Irrelevant functionalities

Teachers found no functionality irrelevant.

3.1.2.3 Students

Students' reactions were mixed. The system was not rated negatively; some only questioned its use and teacher activity. Seven out of thirteen students would use the profile.

Functionalities to add

- Choosing your own skills

Description: “Data that the student selects can be more relevant than the system-generated data. I'm not going to list Java as my skill if I want to work in C++.”

Response Section “Subjective proficiency” with the ability to define own skills will be added.

Irrelevant functionalities

Students found no functionality irrelevant.

3.1.3 Summary of requirements

After reaching out to all of the participating groups, the list of all selected functional requirements was sorted by priority of individual requirements, followed by a list of non-functional requirements.

3.1.3.1 Functional requirements

- F1 – Profile filtering

Priority: high

Description: The system will allow filtering of profiles by parameters. Parameters of filtering will be skills, study year and programme, availability status, interest in the study field, and professional experience. Only the profiles visible to the user, based on his roles, appear in the filter results.

- F2 – Generating skills from grades

Priority: high

Description: The system will collect data from Grades during scheduled intervals and convert them into skill points.

- F3 – Skill endorsements

Priority: high

Description: The system will record students' skills endorsements. It will allow both adding and revoking of endorsements, which can be done only by teachers.

- F4 – References

Priority: high

Description: The system will record students' references. It will allow adding, editing, and removing references, which can be done only by teachers.

- F5 – Subjective proficiency

Priority: high

Description: The system will record students' subjective proficiency. It will allow adding, editing, and removing of proficiency. Only students can set it to themselves.

- F6 – Profile visibility

Priority: high

Description: The system will record profile visibility to allow students to show their profile only to a specific group of users by role. Only students can set visibility to themselves. The student's profile will be hidden to all groups by default and student has to change this setting explicitly to become visible for filtering.

- F7 – Searching profiles by name

Priority: high

Description: System will allow to search for profiles by name. Only the profiles visible to the user, based on his roles, appear in the search results.

- F8 – Roles

Priority: high

Description: The system will record user roles. There will be three types of roles: student, teacher, company.

- F9 – Professional experience

Priority: medium

Description: The system will record professional experience. It will allow adding, editing, and removing of experience. Only students can set it to themselves.

3. ANALYSIS

- F10 – Interest in the study field

Priority: medium

Description: The system will record student's interest in the study fields. Students will receive a fixed amount of points, they can distribute among all study fields taught at FIT CTU. The more points they assign to a field, the more they are interested in it.

- F11 – Availability status

Priority: medium

Description: The system will record students' availability status and collaboration types. Among the types of collaboration will be, for example, master thesis, school project, part-time etc. Availability status is automatically set to available/unavailable based on the requested collaboration. If the student only wishes to be visible, but not receive any offers, then having status unavailable with the profile set to visible is the solution.

3.1.3.2 Nonfunctional requirements

- N1 – Technology stack similar to Grades

Priority: high

Description: As the Skills system will be developed by the Evolution team in the future, it is required to use technology stack similar to the stack of Grades. Used technologies will be in their latest versions to provide the longest possible support. Therefore versions may vary.

3.2 Use case modelling

This section focuses on the actors and the use case modelling.

3.2.1 Actors

Student A studying person, who is responsible for managing their profile.

Teacher A person, employed at the school, who is responsible for giving references and giving skill endorsements.

Company An external person, who is authorized to search among students.

3.2.2 Main use cases

Main use cases and their actors are shown in Figure 3.3. Table 3.1 shows that all functional requirements are covered by use cases.

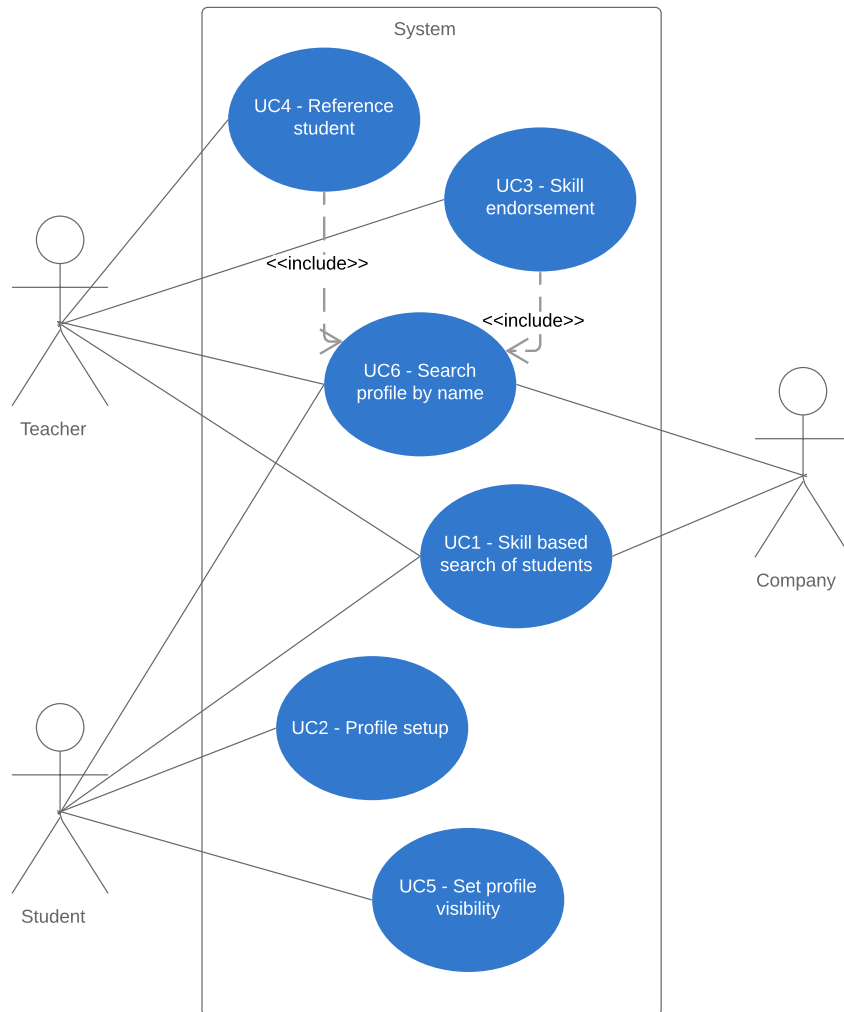


Figure 3.3: Use case diagram

3. ANALYSIS

UC1 – Skill based search of students

Description: Allows searching for students based on specified criteria.

Actors: Any.

Steps:

1. The use case starts when a logged in user wants to search for students by criteria.
2. The user selects the filtering criteria.
3. The user presses the search button.
4. The system displays a list of matching results.

UC2 – Profile setup

Description: It allows students to customize their profile to reflect current skills and interests.

Actors: Student

Steps:

1. The use case starts when the logged in user wants to edit the information on his profile.
2. The user selects the option to edit the desired section.
3. The user makes the changes.
4. The user presses the save changes button.
5. The system displays the edited profile.

UC3 – Skill endorsement

Description: Allows teachers to add or revoke an endorsement of student's skill.

Actors: Teacher

Steps:

1. Include UC6 – Search profile by name.
2. The user presses the endorse a skill button.
3. The system displays a window with the option to add a comment.
4. The user presses the save button.
5. The system stores the endorsement.

UC4 – Manage reference

Description: Allows teachers to add or remove a student's reference.

Actors: Teacher

Steps:

1. Include UC6 – Search profile by name.
2. The user presses the add reference button.
3. The system displays a window with a required field to fill in the reference.
4. The user fills in the field.
5. The user presses the save button.
6. The system stores the reference.

UC5 – Set profile visibility

Description: Allows students to show or hide their profile.

Actors: Student

Steps:

1. The use case starts when the logged in user wants to change the profile visibility.
2. The user navigates to the settings page.
3. The user changes the visibility for a certain role by pressing the toggle button.
4. The user presses the save button.
5. The system stores the settings.

UC6 – Search profile by name

Description: Allows user to search for profiles by name.

Actors: Any.

Steps:

1. The use case starts when a logged-in user wants to search for a profile by name.
2. The user enters the student's name into the search field on the main bar.
3. The system offers a list of profiles with a matching name.
4. The user presses the list item.
5. The system navigates the user to the selected profile.

3. ANALYSIS

	UC1 – Skill based search of students	UC2 – Profile setup	UC3 – Skill endorsement	UC4 – Manage reference	UC5 – Set profile visibility	UC6 – Search profile by name
F1 – Profile filtering	✓					
F2 – Generating skills from grades	✓					
F3 – Skill endorsements	✓		✓			
F4 – References	✓			✓		
F5 – Subjective proficiency	✓	✓	✓			
F6 – Profile visibility	✓				✓	✓
F7 – Searching profiles by name				✓		✓
F8 – Roles	✓	✓	✓	✓	✓	✓
F9 – Professional experience	✓	✓				
F10 – Interests	✓	✓				
F11 – Availability status	✓	✓				

Table 3.1: Functional requirements realisation

Architecture

The chapter focuses on realising non-functional requirements. It describes the division into a client and a server and describes each side.

4.1 Client-server architecture

Inspired by the design of Grades, system will make use of the Client-server architecture as shown in Figure 4.1, where all the data manipulation is handled by the server side (from now referred to as the Server) and data presentation is handled by the client side (from now referred to as the Client). This approach ensures that possible mistake, done during development of the Client, has no chance of promoting to the Server, due to separate development.

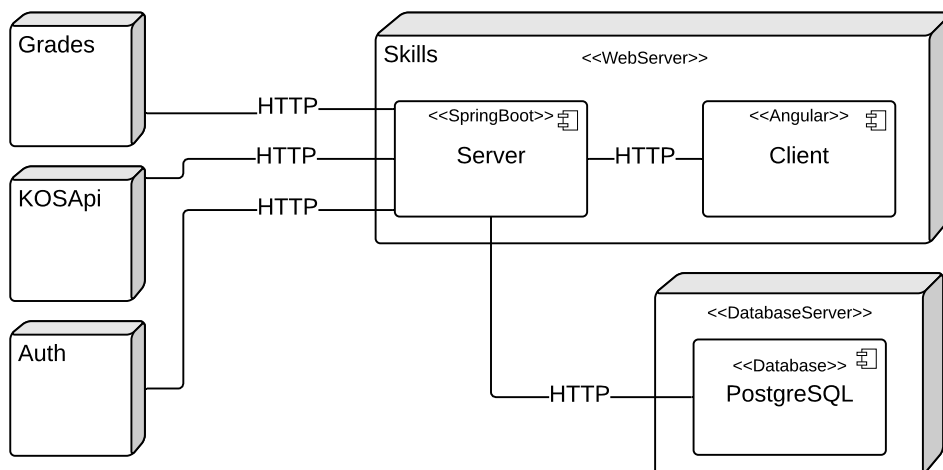


Figure 4.1: Deployment diagram

All communication will be provided through the REST interface (HTTP). To receive data for generating skill points, Server has to communicate with Grades (discussed in Section 5.2 and Section 6.2.1). KOSApi will provide user's information and roles (discussed in Section 6.4.2). Authorization is done via the FIT Authorization server (discussed in Section 6.4).

4.2 Server-side architecture

As was already mentioned in Section 2.3, Layered architecture is the most fitting for this project. The following sections describe division into layers and code structure.

4.2.1 Architecture description

The Server is divided into four layers: Web, Service, Data, Database. Each layer has its own responsibilities. Furthermore, in Figure 4.2, each layer is labelled *closed*, which is based on the Open-closed layer principle. In the case of a closed Service layer, this means that any requests received on the Web layer must pass through the Service layer to reach the Data layer. The reason for this solution is the concept of layers of isolation which means that changes to one layer generally do not impact or affect components in other layers.

Web Layer

Web layer contains REST controllers with HTTP endpoints. These endpoints define the Server's API (Application programming interface). User input can be received through URL parameters or a DTO (Data Transfer Object). Once a request is received, *input validation* is done by Spring to keep input sanitized. After input validation, *privilege check* is run to verify that the user is allowed to use the requested endpoint. A validated request is then delegated to the Service layer for further processing.

All data flowing in and out of REST controllers are deserialized and serialized into JSON by Spring automatically. Exception handling is also one of the responsibilities that spring can handle. All delegated operations can throw exceptions and for such cases an `ExceptionHandler` is set up to convert exceptions into proper HTTP response codes and messages.

Service Layer

Service layer contains what is called a *business logic* of the application. Such logic holds system specific restrictions such as only a person with the role Teacher can give reference to a profile. This layer also manages operations in a transaction-like manner, which means that either all operations of a service

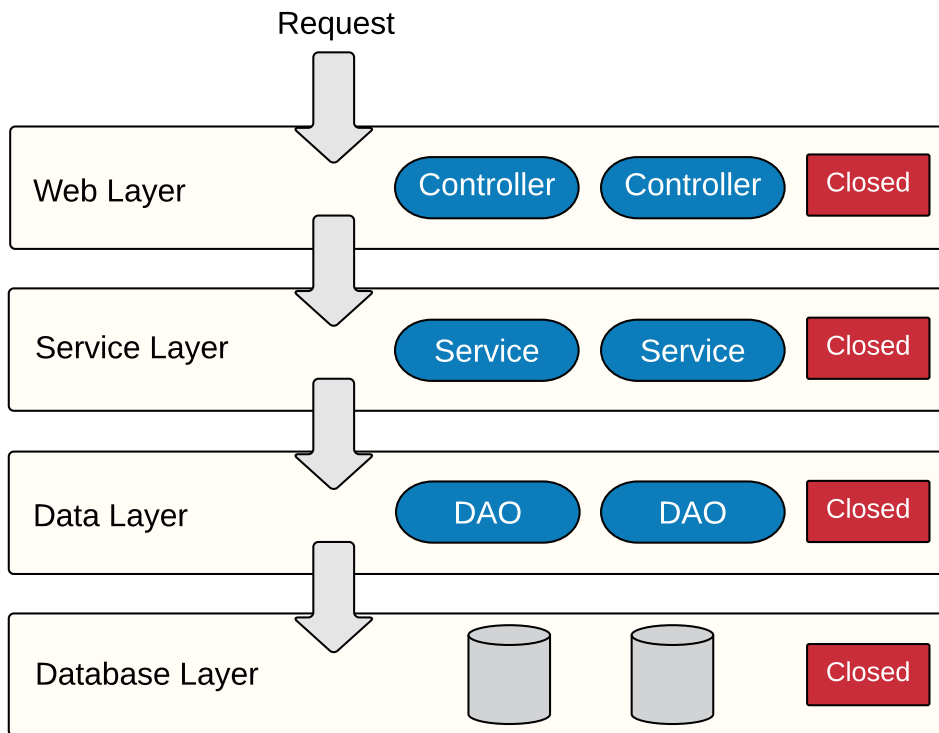


Figure 4.2: Layered architecture. Source: [35, edited by the author]

method succeed or every change is rolled back. A transaction is especially useful for a method calling multiple data layer methods.

Since all data is already sanitized from the Web layer, no input validation needs to be done, but if a service method is expected to return something other than void, then it is Service layer's responsibility to *convert processed data* to a DTO. DTO pattern can be used to hide entity column names to prevent SQL attacks but more than that it is widely used to return data matching a specific use case. If all a method requests is a person's username, then giving away other attributes such as a person's age would be redundant.

Data Layer

Data layer provides simplified access to data stored in a persistent storage. Once a data source is configured via Spring `@Configuration` an `EntityManager` bean is available to manage and search entities in the relational database, and each of Data layer's public method represents one database transaction. Spring allows switching between configurations by assigning a specific profile to each configuration. Registered profiles are: "test" for testing, "dev" for

development and “prod” for production. Setting up configuration this way allows to quickly switch between databases when necessary.

Since databases are not the only source of data, sources like Grades and KOSApi are in a separate package. Each package contains domain models of the corresponding domain and DAO interfaces for data manipulation. Package distribution is following:

internal – Manages data of internal databases.

external – Manages data of external sources, e.g., Grades.

Database Layer

Contains configurable data sources for data persistence. Three PostgreSQL data sources are used:

DB – Database containing all internal data.

DW – Data warehouse, which contains precalculated data for faster queries.

Test – Database used for testing.

4.2.2 Code structure

The code is structured into modules, which are coupled together with module dependencies, as shown in Figure 4.3. Open-closed layer principle is enforced through maven dependency management, which allows disabling of transitive dependencies. Modules’ purposes are following:

skills-web – Represents Web layer.

skills-service – Represents Service layer.

skills-data – Represents Data layer.

skills-comms – Contains classes shared across all server modules (e.g., DTOs, exceptions, enums, ...).

skills-boot – Contains main class for starting Spring Framework, which wires all layers together.

skills-ui – Client application.

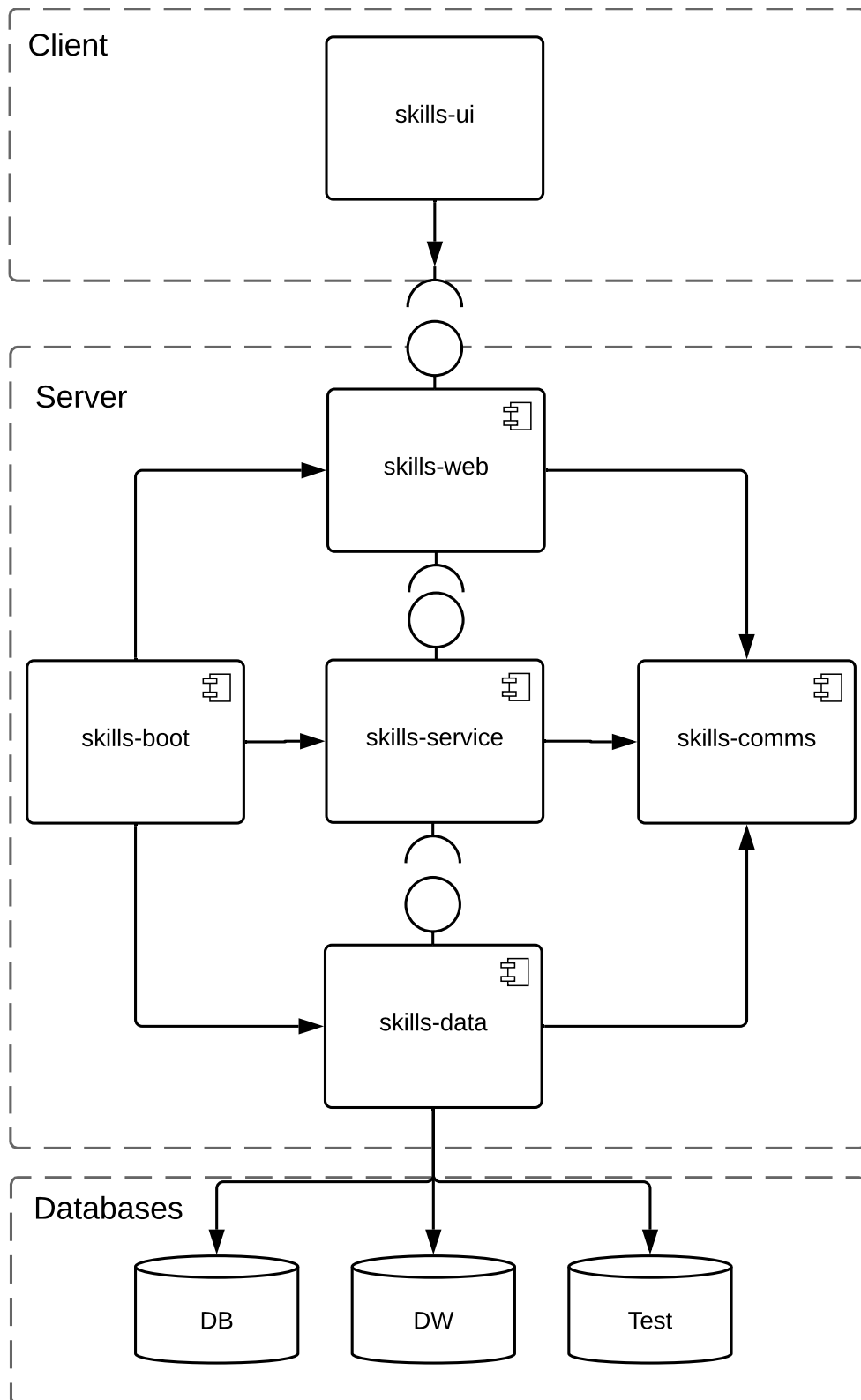


Figure 4.3: Module dependencies

4.3 Client-side structure

For easier navigation through the Angular project, files are grouped into folders based on their purpose. The following is a code structure of the `skills-ui` module:

views – Each view represents one screen page.

services – Services define logic, perform HTTP requests or share data.

resolvers – Resolvers call services to fetch data from server. Attaching a resolver to a page blocks the page. from displaying until data is ready.

components – Small reusable elements, which can be combined together to build a view.

models – Classes, enums, DTOs.

dialogs – Components displayed as modal dialogs.

Design

This chapter focuses on the realization of functional requirements. Screen prototypes will be introduced along with a description of the skill points calculation process and the search algorithm. Last section describes design of database model.

5.1 Profile

Profile layout is designed in a simple two-column layout with a wide block of skills at the bottom as can be seen in Figure 5.1.

Profile sections went through several changes during the design process. A major change occurred in the generated skill set section, where the initial multilevel pie chart had been replaced with a skill cloud because there are no data yet to support the skill hierarchy required for the pie chart. Multilevel pie chart still stands as a proposed feature to improve readability of the skill set, but working on a structure supporting this hierarchy, would exceed the thesis' scope. To make a profile more interesting for the users, new sections are introduced, such as "Professional experience" and "Status", as requested by the potential users.

To reflect what fields and technologies is a student interested in, sections "Subjective proficiency" and "Interests" were added. This decision is justified by the fact that most people are looking for students with a great interest in the respective field and ability to learn, not necessarily students with a certain level of skill.

5. DESIGN

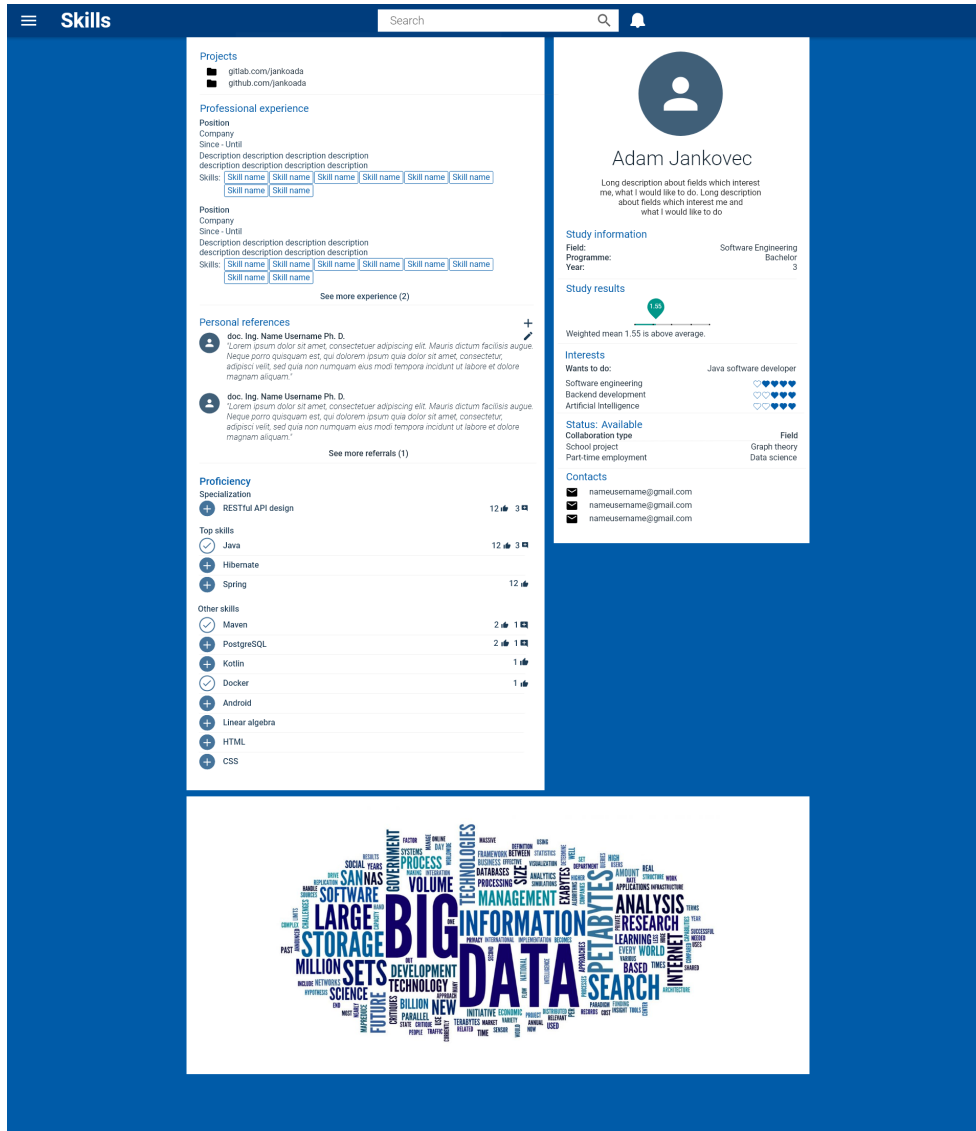


Figure 5.1: Profile prototype from teacher's perspective

5.1.1 Sections

Introduction

This section (shown in Figure 5.2) contains a profile photo and a student's full name. A student can also add a brief description of his goals and motivation.

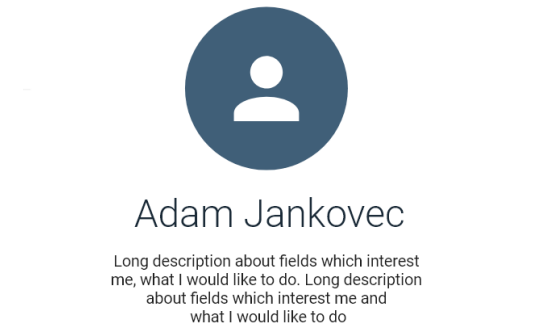


Figure 5.2: Detail of Introduction section

Study Information

This section (shown in Figure 5.3) provides basic information about user's study such as study field (e.g., Software Engineering), year, and programme (e.g., Bachelor). Data is available from KOSApi.

Study information

Field:	Software Engineering
Programme:	Bachelor
Year:	3

Figure 5.3: Detail of Study Information section

Study results

This section (shown in Figure 5.4) provides a weighted mean of student's results. Data is available from Grades' API.

Study results

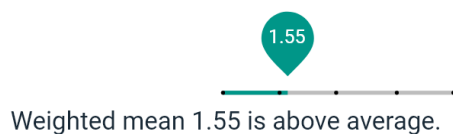


Figure 5.4: Detail of Study results section

Interests

This section (shown in Figure 5.5) targets students' interests with a field, where they can fill out their dream job. Students also receive a limited amount of points to distribute among all IT (Information Technology) fields. The higher the amount of points field receives, the more student is interested. A maximum amount of points to assign to one field is 5.



Figure 5.5: Detail of Interests section

Status

Students can set types of collaboration they are interested in. There are several types of collaboration such as bachelor thesis, master thesis, part-time employment, semester assignment, etc. Section is shown in Figure 5.6.



Figure 5.6: Detail of Status section

Contacts

This section (shown in Figure 5.7) allows students to enter means of communication, through which they are available to discuss collaboration.

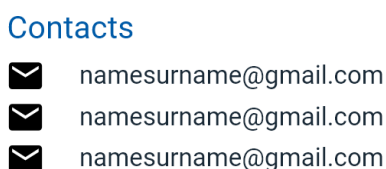


Figure 5.7: Detail of Contacts section

Projects

This section (shown in Figure 5.8) allows students to provide links to their projects on sites such as GitLab, GitHub, etc.

Projects

- gitlab.com/jankooda
- github.com/jankooda

Figure 5.8: Detail of Projects section

Professional experience

This section (shown in Figure 5.9) allows students to enter their professional experience. Each experience contains job position, company name, dates “since” and “until”, description, and skills. Listed skills will provide an opportunity to refine the filtering of profiles by skills.

Professional experience

Position

Company

Since - Until

Description description description description

description description description description

Skills: Skill name Skill name Skill name Skill name Skill name Skill name
Skill name Skill name

Figure 5.9: Detail of Professional experience section

References

This section (shown in Figure 5.10) contains references received from teachers. Teachers have an option to add multiple references, as well as edit and delete their references. Each reference contains author photo, name, and comment.

References



doc. Ing. Name Surname Ph. D.

"Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris dictum facilisis augue. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam."



Figure 5.10: Detail of References section

Subjective proficiency

This section (shown in Figure 5.11) allows students to define their strengths. Proficiency consists of three lists: specializations, main skills, other skills. Proficiency helps refine filtering profiles by skills. Having a visible, empty specialization field could also motivate students to choose an area they could specialize in.

Teachers can endorse these fields, with an optional comment to specify the reason behind the endorsement. The button to the left indicates that a teacher can either give or revoke given endorsement. Icons to the right indicate how many endorsements were given, and clicking the icon opens a list of endorsement authors and comments.

Subjective Proficiency

Specialization

+ RESTful API design 12  3 


Top skills

✓ Java 12  3 

+ Hibernate

+ Spring 12 

Other skills

✓ Maven 2  1 

+ PostgreSQL 2  1 

+ Kotlin 1 

✓ Docker 1 

+ Android

Figure 5.11: Detail of Subjective proficiency section

Skill cloud

This section (shown in Figure 5.12) shows all student's skills generated from Grades. Each skill has a certain size based on the number of points student collected for that skill. To ensure readability, sizes are normalized to a fixed range of font sizes.



Figure 5.12: Detail of Skill cloud section. Source: [36]

5.2 Skill points calculation

Generating skill points uses data collected from Grades. Each task, test, and course student completes, already has assigned skills and weight. After designing the endpoints and DTOs for exporting data from Grades, processing was designed next.

The whole process depicted in Figure 5.13 begins with fetching of all courses with defined skills. Since collected data is expected to grow with time, measures have to be applied to account for such expectation. Not fetching data all at once, but only instance by instance, significantly reduces memory impact on the system. Courses contain defined skills and instances' semester codes. Each semester code, paired with course code, is then used to fetch its course instance. All instance's evaluations are processed into skill points with the following evaluation.

$$\text{points} = \text{creditsCount} \cdot \text{skillWeightCoefficient} \cdot \text{evaluationPercentile}$$

creditsCount Number of credits gained for completing a course. This means that students gain more skill points from harder courses.

skillWeightCoefficient Skill weight is defined by teachers in Grades to put more emphasis on main taught skills rather than the supplementary skills. Coefficient ranges from 0 to 1.

evaluationPercentile Course difficulty can change over time. One semester, there can be 50% failure rate, the other only 20% rate. Calculating points by evaluation percentile and not by evaluation value takes these differences into account and allows for a fair points distribution among all semesters. Percentile ranges from 0 to 1.

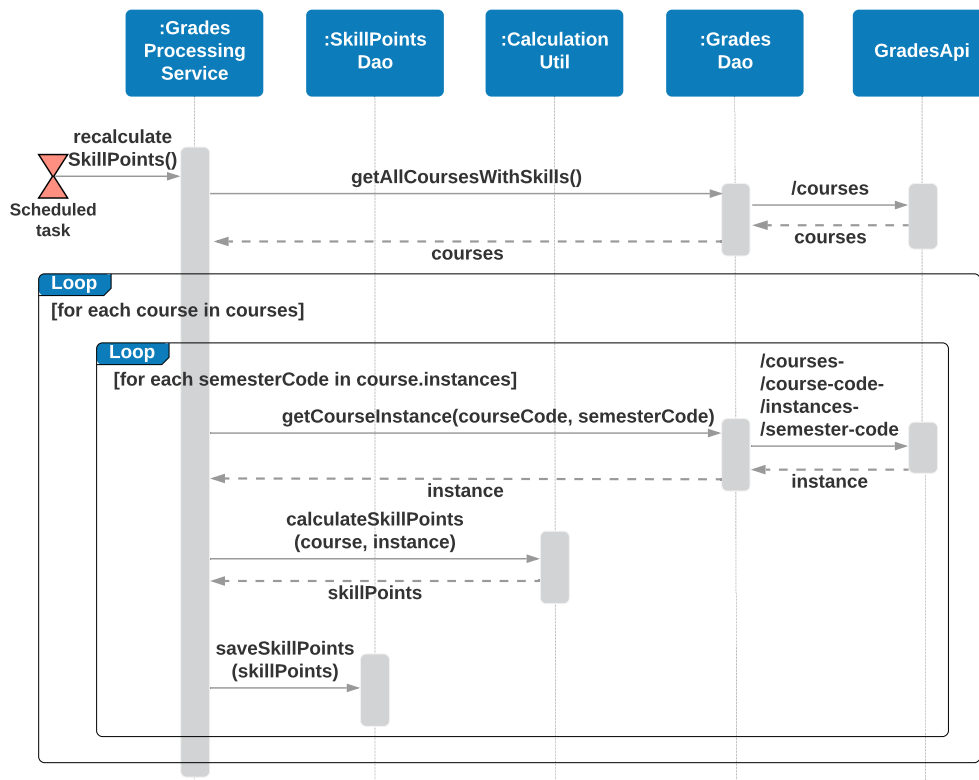


Figure 5.13: Sequence diagram of processing grades into skill points

All calculated points are then persisted in batches via `SkillPointsDao`. Scheduler, managed by Spring, automatically triggers the process of recalculating skill points every 24 hours, followed by a global skill points calculation.

Global skill points calculation consists of creating a sum of points for each skill across all students. Sums are stored in DW (data warehouse) database, prepared for a fast retrieval during global skill cloud setup when using advanced search shown in Figure B.8.

5.3 Searching for profiles

One can access the user's profile either through the search bar or the filter page. Both ways still respect visibility settings of profiles so they will appear in the search results only with the owners' consent.

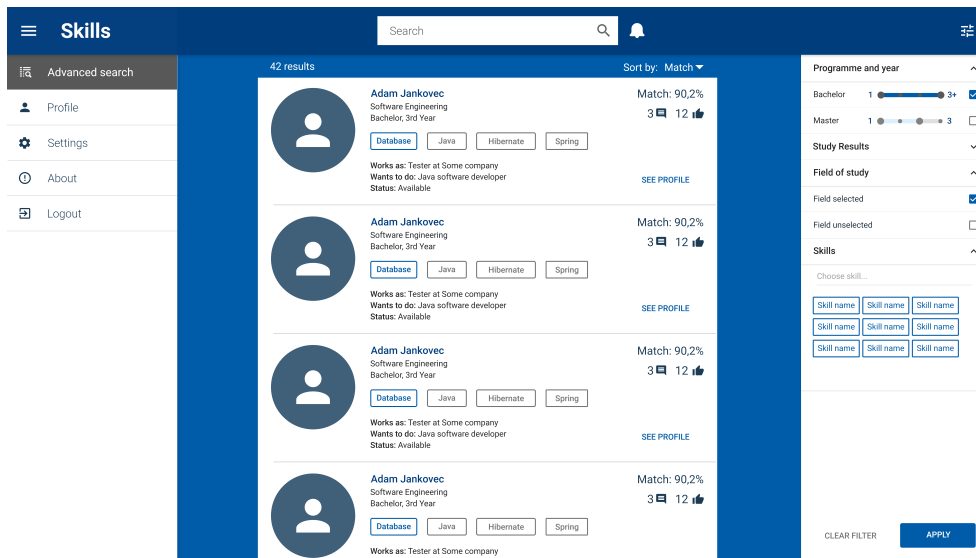


Figure 5.14: Search page prototype

5.3.1 Search by name

The search bar is a text input field equipped with an auto-complete function. It is placed at the top bar, as illustrated in Figure 5.14. Whenever a user enters text, auto-complete is triggered, and a request is sent to the Server. The text is compared with the full names of people in a case-insensitive manner. Results appear in a dropdown list with clickable links to profiles.

5.3.2 Advanced search

Search parameters

Search parameters are depicted in the prototype shown in Figure 5.14, and consist of two types of parameters: “fixed” and “flexible”. Filtering based on fixed parameters will filter out any non-matching profiles, unlike flexible parameters, which only decrease the match percentage of a non-matching profile. Parameters to filter by are following:

- Flexible parameters

Skills Skills, which were generated or manually added by students as subjective proficiency.

Study field Enrolled fields, taught at FIT CTU, such as Security, Data Science, etc. This field was moved to flexible parameters since students at FIT CTU are allowed to study without having their field selected.

- Fixed parameters

Programme & Year Ranged slider for year of study and checkboxes for active study (e.g., bachelor, master).

Study mean Ranged slider of acceptable weighted study mean ranging from 1.0 to 2.5.

Endorsements & References Ranged slider representing number of endorsements and references helps filtering out non-active profiles.

Professional experience Ranged slider representing amount of professional experience allows to specifically filter experienced students

Status Allows binary swapping between “available” and “unavailable” to filter out profiles open for cooperation.

Collaboration type Checkboxes with collaboration types (e.g., master thesis, full-time, any, ...).

Search algorithm

Implementing a sophisticated search algorithm would exceed the thesis’ scope, so only naive skill-based filtering is designed and implemented to provide basic support for the Proof of Concept.

Algorithm 1 Profile evaluation

Require: a set of *profileSkills* to evaluate, a set of *requestedSkills* to filter

Ensure: a pair of (*match*, *points*)

- 1: $matchingSkills \leftarrow intersect(requestedSkills, profileSkills)$
 - 2: $match \leftarrow |matchingSkills| / |requestedSkills| \cdot 100$
 - 3: $points \leftarrow 0$
 - 4: **for all** $skill \in matchingSkills$ **do**
 - 5: $points \leftarrow points + skill.points$
 - 6: **end for**
-

After gathering profiles which are visible to the user, each profile is then evaluated as illustrated in Algorithm 1, which shows the calculation of match and sum of matching skills. All evaluated profiles are then sorted in descending order first by *match* then by *points* to ensure that search favours profiles with higher a sum of points when matching skill sets are found.

Before returning the results, final check is ran to find if any results match the requested skills. If the *match* value of the person with the highest *match* equals to zero, it means that no profile is relevant for the query and a message appears to the user telling that no results were found.

Listing search results

Search results appear in a vertical list as illustrated in Figure 5.14. Navigation to a profile is available through clickable avatar, name or “see profile” button.

Each list item contains profile preview, which provides a subset of profile information to let the user decide whether the profile is relevant to his request. The preview contains general information such as full name, field of study, programme and year of study. Under the general information are user’s specializations and main skills followed by current employment labelled as “Works as” and possible dream position labelled as “Wants to do” underlined with availability status. Match relevance is displayed in percentage supplied by a number of endorsements and references to inform about profile activity.

5.4 Database model

During the database design process (also discussed in Section 2.2), only the features that were selected for implementation were included. Adding features, which could be discarded in the future, would only add redundant complexity, apart from that, the design will be used by the Grades team as a documentation of the implemented solution. Proposed profile extensions are independent of each other, therefore extending the design is possible by simply connecting new tables without affecting the existing tables.

To create a database schema mapped from the Server, two approaches were considered. The first was making a database create script, and then mapping tables to Java classes with ORM. The second was creating Java classes annotated as entities and specifying restrictions with annotations, and then through the ORM letting Hibernate generate the schema. For the sake of simplicity, the latter approach was chosen. However, the approach with a create script is more suitable for database versioning and should be adopted when the project is handed over to the Evolution team.

Schema generated with Hibernate, illustrated in Figure 5.15, conforms the 3NF and comprises of the following tables:

Person Represents users of the system and contains their personal information such as their name and username.

Skill Represents a skill and contains its name.

PersonSkill Association table decomposed from an M to N relationship between Person and Skill. Contains skill points calculated from grades.

Endorsement Represents skill endorsement and contains its date of submission, author, and comment.

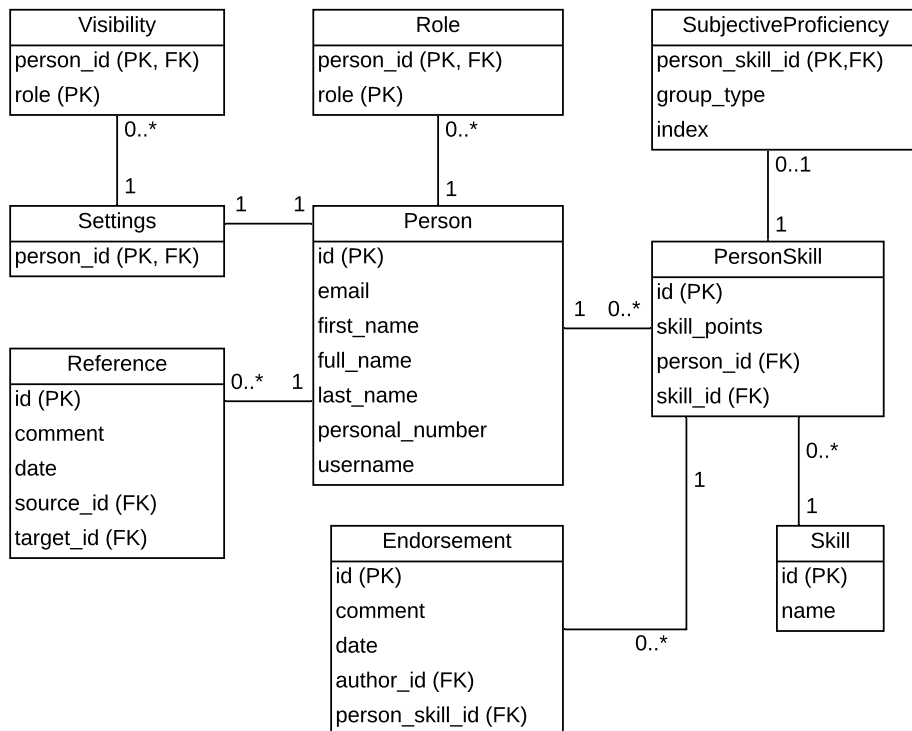


Figure 5.15: Database model

SubjectiveProficiency Represents subjective proficiency with group type enum value (Specialization, Main, Other) and index, which specifies the order of skills in the list.

Reference Represents reference and contains the author as the source and the receiver as the target, date of submission and comment.

Role Represents user's roles and contains a role field with enum value (Teacher, Student, Company). Presence of the row means the user has the specified role.

Settings Represents user's settings.

Visibility Represents user's profile visibility and contains a role field with enum value same as Role table which allows for easy database queries to seek visible profiles based on user's roles. Presence of the row means the user is visible to the specified role.

Implementation

This chapter contains a summary of the requirements chosen for the implementation of the Proof of Concept. It describes the process of implementation of the Client and the Server, and steps taken to overcome obstacles that were encountered during development. The Security section dives into authorization using industry-standard protocol OAuth 2.0.

6.1 Requirements

Among the requirements, chosen for the implementation part of the thesis, are only those with the highest priority, which are as follows:

- F1 – Profile filtering,
- F2 – Generating skills from grades,
- F3 – Skill endorsements,
- F4 – References,
- F5 – Subjective proficiency,
- F6 – Profile visibility,
- F7 – Searching profiles by name,
- F8 – Roles.

6.2 Server-side development

Setting up a project with Spring Boot requires minimal effort due to the provided auto-configuration, which is available through Spring-Boot-Starter dependencies. However, setting up this project proved to be quite demanding,

method	name	URI
GET	getAllCourses	'/courses'
GET	getInstance	'/courses/course-code/instances/semester-code'
GET	getAllSkills	'/skills'
GET	getUserInfo	'/users/username/info'
GET	getUserRoles	'/users/username/roles'

Table 6.1: REST endpoints implemented in Grades

because a multi-module project with limited visibility for each module creates lots of inconveniences because it requires a deeper knowledge of how Spring's configuration and lookup works. Development faced many challenges with unsatisfied dependencies for dependency injection, because of module visibility, but after carefully re-reading Spring documentation, all the issues were resolved and the resulting project could pose a solid foundation for the future projects.

6.2.1 Getting data from Grades

Grades, as an external source of data, has its place on the Data layer in package `external`, with an assigned component named `GradesDao`. There were two possible ways to handle the data source. One by mocking the `GradesDao` object, which would take some time to simulate the usual flow of data, that would be received. The other by implementing the endpoints right into the source code of Grades. Even though altering Grades' code would be out of scope, author is one of the developers with access to Grades and its data. Preparing Grades' code for the upcoming changes would also push this project closer to deployment, therefore, the decision was made to alter the source code of Grades.

Support for skills was already implemented in Grades, so the only thing left to code was mapping the data to the requested DTOs and coding a REST API to share data with the Server. There were still a few changes, which had to be done to the Grades' database, in order to get the required data. Coding this support set the project back by one week, which was a good trade-off for a stable connection to the source of data, used during the implementation phase. Implemented endpoints can be seen in Figure 6.1.

6.3 REST API

The Client communicates with the Server via REST API, which is defined by the Server. Each endpoint represents one use case and has specific parameters and response codes. All methods serialize data into JSON format, and their descriptions are provided in Appendix C.

Requests by unauthorized users receive response with status code 401 (Unauthorized). Requesting resource with insufficient privileges returns status code 403 (Forbidden). Erroneous requests (e.g., non-existent resources) result either in 400 (Bad request) or 404 (Not Found).

6.4 Security

Data security is one of the critical parts of the system. To ensure consistency with the Grades project, security measures mimic those used in Grades, which comprise of the OAuth 2.0 protocol and the FIT authorization server.

6.4.1 Authorization

Authorization of users is done through the FIT authorization server. One has to register an application in the authorization server manager [37] before sending authorization requests.

Spring provides a module called Spring Security which requires minor configuration for setting up authorization. `Client-id` and `client-secret`, received from the manager, are pasted into a server configuration file, and Spring Security handles automatically the creation of a `/login` endpoint which, when triggered, starts the whole process of authorization.

Whenever an unauthorized user requests a resource, the Client asks the Server if the user is authorized. Authorized users will be granted permission to access the requested resource. Otherwise, the user is redirected to login page to start the authorization process illustrated in Figure 6.1.

6.4.2 User roles

There are three types of roles: student, teacher, company. Students and teachers are fetched from Grades, unlike companies, whose accounts are fetched from KOSApi. Since roles of users may change anytime, user roles are refreshed on every user sign in, to reflect the changes of roles immediately.

Role privileges and purposes were previously described in Section 3.2.1. Another purpose of these roles is to allow students to decide who can access their profile. A profile is hidden to everyone by default and student has to manually change the setting to allow access (as shown in Appendix B.10). This prevents an accidental display of user's information without the user's consent, while allowing to select which roles can access user's profile.

To enforce user privileges, each REST endpoint is secured through a Service layer component `SecurityService`, which provides methods to check user's privilege.

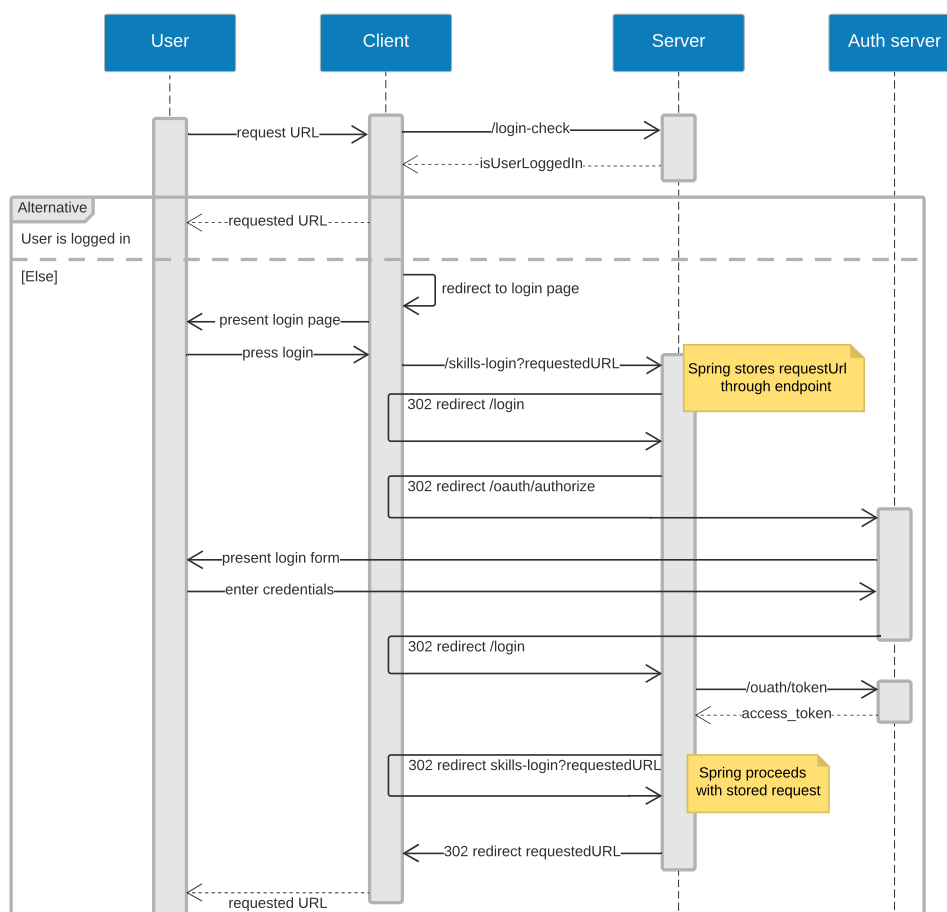


Figure 6.1: Authorization flow

6.5 Client-side development

Since AngularJS (used in Grades) and Angular 7 are vastly different, the initial setup had to be completely redone. Even though all screens were captured in detail by the prototype, final implementation still differs in details like colour and font sizes, but layout remains the same.

Since UI (User Interface) responsiveness was taken into account during prototyping, coding the responsive UI was straightforward. However, having a majority of components pre-written with Angular Material caused some minor setbacks, because of a lack of flexibility when altering the CSS. These setbacks were balanced by the fact, that writing these components from scratch would delay the project by at least two months even if the final design was at disposal.

6.5.1 Resolving CORS policy violation

A problem occurred during an attempt to fetch data from the Server to the Client. It was caused by the browser, which was blocking any request, due to CORS (Cross-origin resource sharing) policy violation. The violation is triggered whenever a client requests a resource from another domain; in this case, the Client and the Server were running on different ports of localhost, which occurs only in development environment. In production environment, where client and server are deployed to the same domain, CORS violation is no longer triggered.

Two solutions were available to resolve this issue. One by adding a “Access-Control-Allow-Origin” header to each request, which automatically allows anyone from anywhere to ask for a resource and thus sacrificing the ability to specify which domain can access the Server’s resources. The other option was to add a proxy to redirect requests from the Client, which leads to keeping the ability to specify, who can access the resource. Since Angular already has built-in support for such proxy, it was decided to choose the latter option.

The code example in Figure 6.2 shows how easy it is to setup such proxy with Angular. Any request prefixed by “/api” will be prefixed by the contents of the *target* attribute. Figure 6.3 illustrates how proxy translates the requests.

```
{
  "/api": {
    "target": "http://localhost:9002/",
    "changeOrigin": true,
  }
}
```

Figure 6.2: Proxy configuration

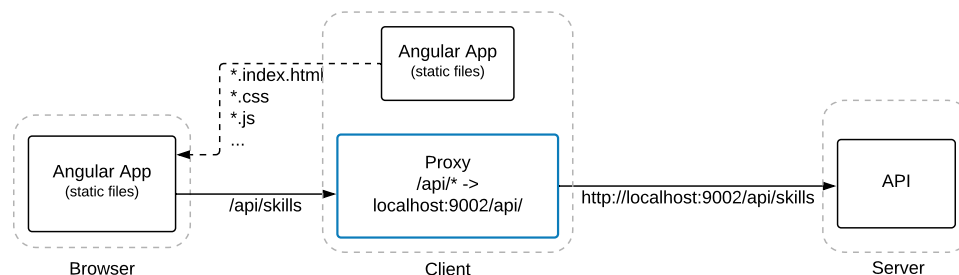


Figure 6.3: Request translation via Proxy. Source: [38, edited by the author]

Testing

This chapter describes the measures, which were applied to reveal any existing bugs and mitigate the chances of introducing new bugs in future development. It also describes usability testing and acceptance testing.

7.1 Automated testing

Separation of each layer into its reserved module allows each layer to have its own set of tests. Mocking is used to isolate the tests from other layers. Thanks to the Open-closed principle (discussed in Section 4.2), each layer requires only mocking of the layer underneath, for example, testing the Service layer requires only mocks of the Data layer. This is one of the strongest points of the Layered architecture.

The Server is mostly a CRUD (Create Read Update Delete) application, so most methods contain little to no business logic, and testing them would have a low change of finding any major bugs, therefore, only the critical parts of the system were identified for testing. Perfecting the whole Data layer and its handling of internal and external sources will stand as a good foundation for future development. Data layer testing is done with DbUnit, which is a JUnit extension targeted at database-driven projects. DbUnit can set the database into a known state between test runs to avoid scenarios, where one test case corrupts the database and causes subsequent tests to fail. Database data is loaded from XML datasets, which are verified after the manipulation, to match an expected dataset of values. [39]

Figure 7.1 illustrates an example of a Data layer test. This specific example tests a removal of existing skill from the database. `@DatabaseSetup` annotation, with a dataset depicted in Figure 7.2, is used before a test to form the database into a required state. After a test execution, `@ExpectedDatabase` annotation asserts whether the required dataset matches the current database state or not.

7. TESTING

```
@Test
@Transactional
@DatabaseSetup("dataset/SkillDao/deleteSkillPre.xml")
@ExpectedDatabase(
    value = "dataset/SkillDao/deleteSkillPost.xml",
    assertionMode = DatabaseAssertionMode.NON_STRICT_UNORDERED
)
public void deleteSkillById() {
    Long idOfExistingSkill = 1L;
    skillDao.deleteSkill(idOfExistingSkill);
}
```

Figure 7.1: Example of testing database transaction with DbUnit

```
<?xml version="1.0" encoding="UTF-8"?>
<dataset>
    <Person id="1" username="person1"/>
    <Person id="2" username="person2"/>
    <Skill id="1" name="Java"/>
    <Skill id="2" name="Spring"/>
    <Person_skill id="1" person_id="1" skill_id="2"
        skill_points="30"/>
    <Person_skill id="2" person_id="1" skill_id="1"
        skill_points="20"/>
    <Person_skill id="3" person_id="2" skill_id="1"
        skill_points="20"/>
</dataset>
```

Figure 7.2: Example of dataset consumed by DbUnit

```
...
@Test(expected = GradesApiUnavailableException.class)
public void givenCallToGetUserInfo
    _whenGradesOffline
    _thenThrowGradesApiUnavailableException() {
    ...
    mockServer.expect(requestTo(url))
        .andRespond((response) -> {
            throw new ResourceAccessException(...);
        });

    gradesDao.getUserInfo(...);
}
```

Figure 7.3: Example of mocking an external data source

Testing of external data sources is done via mocks. After manual check of the Grades' API behaviour for each type of request, mocks were written for these scenarios to make sure that Server reacts accordingly. Example of mocking the Grades' API, shown in Figure 7.3, tests a scenario, where the API is offline, and the system is expected to throw an exception named `Grades-APIUnavailableException`, which is consumed by the `ExceptionHandler`.

Summarized code coverage measured with IntelliJIDEA IDE can be seen in Figure 7.1. Low method coverage is caused by untested getters and setters of the domain models.

Class	Method	Line
82 %	65 %	79 %

Table 7.1: Test coverage of the Data Layer

7.2 Static code analysis

“Static code analysis is a method of debugging by examining source code before a program is run. It’s done by analyzing a set of code against a set of coding rules. This type of analysis addresses weaknesses in source code that might lead to vulnerabilities.” [40]

Static code analyzer checks the code as work is being done on the build and does *“in-depth analysis of where there might be potential problems in the code.”* [40] *“The principal advantage of static analysis is the fact that it can reveal errors that do not manifest themselves until a disaster occurs weeks, months, or even years after release.”* [41]

7.2.1 SonarQube

“SonarQube is an open source platform to perform automatic reviews with static analysis of code to detect bugs, code smells and security vulnerabilities on 25+ programming languages including Java, C#, JavaScript, TypeScript, C/C++.” [42]

“SonarQube provides the capability to not only show health of an application but also to highlight issues newly introduced. With a Quality Gate in place, you can fix the leak and therefore improve code quality systematically.” [43] Report can be generated by a maven plugin which is run by the command `“mvn clean verify sonar:sonar”` from the parent project folder.

A report, generated by SonarQube, is shown in Figure 7.4. The example shows that SonarQube evaluated all categories with `“A”` and detected zero duplicated blocks. Code coverage is low because testing the whole system would take an excessive amount of time, which was already invested in the development of such a big project and would exceed scope of bachelor thesis.

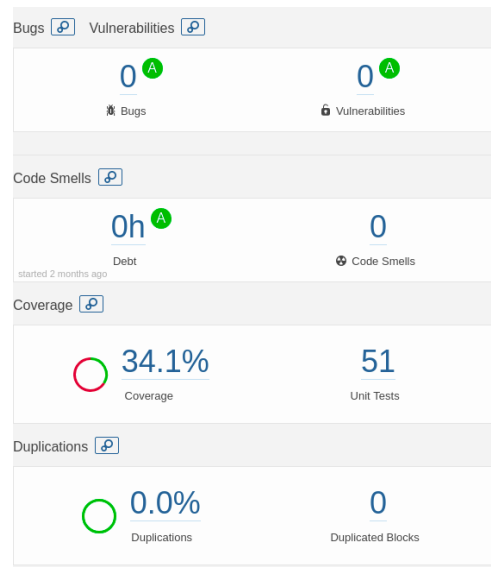


Figure 7.4: Final report generated by SonarQube

7.3 Usability testing

Usability testing refers to testing of the product with representative users. The test consists of a set of tasks, which participants try to complete while being monitored. Participants are often encouraged to comment on their thoughts. The goal is to collect feedback on the product and identify possible weak spots in the UI design, which could lead to confusion of the users. [44]

The process of testing is divided into three test scripts. Each script is specifically designed for its intended group of users, which are students, teachers and companies. Each group has different tasks to complete, based on their privileges in the system. Tasks are then evaluated either as “success”, “success with difficulty” or “failure”. Participants are encouraged to share their thoughts during the testing process and also to give ideas for improvements.

Scripts share the following attributes:

Default state: Search page with logged in user

Goal state: All tasks done

Introduction: We are going to test a system for skill based search of students.

You will be given a set of tasks, try to speak your thoughts and steps out loud. Any suggestions regarding the system improvement would be appreciated.

7.3.1 Test script for companies

Tasks:

1. Add 'java' as one of the skills to filter by.
2. Select one random skill from the Skill cloud as one of the skills to filter by.
3. Apply filter to search for results.
4. Navigate to one of the result's profiles.
5. See who gave endorsement to a skill named 'elasticsearch'.
6. Navigate to a profile named 'Adam Jankovec'.

Results: As can be seen in Table 7.2, task 1 was the only problematic part of the script. The obstacle was caused by an input field for entering a skill name to filter by, which did not appear as an input field to some participants. This problem can be resolved by adding a fitting placeholder as was suggested by one of the participants. The suggestion is documented in Table 7.5.

tasks/companies	company 1	company 2	company 3	company 4
task 1	✓	!	!	✓
task 2	✓	✓	✓	✓
task 3	✓	✓	✓	✓
task 4	✓	✓	✓	✓
task 5	✓	✓	✓	✓
task 6	✓	✓	✓	✓

Table 7.2: Task completion of companies

- ✓ Success
- ! Success with difficulty
- x Failure

7.3.2 Test script for teachers

Tasks:

1. Add 'java' as one of the skills to filter by.
2. Select one random skill from the Skill cloud as one of the skills to filter by.
3. Apply filter to search for results.
4. Navigate to one of the result's profiles.
5. Add a reference to the profile with the text 'test'.

7. TESTING

6. Edit the previous reference to text 'test2'.
7. Delete the reference.
8. Endorse the skill named 'elasticsearch'.
9. See who else gave an endorsement to the previously endorsed skill.
10. Revoke the endorsement of skill named 'elasticsearch'.
11. Navigate to a profile named 'Adam Jankovec'.

Results: Table 7.3 shows task 1 was problematic for most teachers. The teachers noted that the skill input field is hard to spot and instead, they tried to use the search bar for searching by name. This problem has already been addressed in the previous section of testing with companies.

In addition, Teacher 1 noted that adding a trash can as an option to delete reference would be nice, but also noted, that deleting via the edit window is also intuitive. Teacher 2 also mentioned that having a trash can icon to delete the reference would be better because the option to delete it was not expected.

tasks/teachers	teacher 1	teacher 2	teacher 3
task 1	✓	!	!
task 2	✓	✓	✓
task 3	✓	✓	✓
task 4	✓	✓	✓
task 5	✓	✓	✓
task 6	✓	✓	✓
task 7	✓	✓	✓
task 8	✓	✓	✓
task 9	✓	✓	✓
task 10	✓	✓	✓
task 11	✓	✓	✓

Table 7.3: Task completion of teachers

- ✓ Success
- ! Success with difficulty
- x Failure

7.3.3 Test script for students

Tasks:

1. Navigate to your profile.
2. Check what skills were generated for you.
3. Find how many points does your biggest skill have.
4. Navigate to edit subjective proficiency page.
5. Add a random skill.
6. Move the skill to your main skills.
7. Add another random skill.
8. Delete the main skill.
9. Save your new proficiency.
10. Navigate to the search page.
11. Add 'java' as one of the skills to filter by.
12. Select one random skill from the Skill cloud as one of the skills to filter by.
13. Apply filter to search for results.
14. Navigate to one of the result's profiles.
15. See who gave endorsement to a skill named 'elasticsearch'.
16. Navigate to a profile named 'Adam Jankovec'.
17. Navigate to your settings.
18. Set your profile visibility to be visible only by teachers.
19. Save your settings.

Results: Table 7.4 shows that task 6 (Rearranging of subjective proficiency) was problematic for most users, since the row-locking mechanism does not seem intuitive. Student 2 proposed a solution, which was documented in Table 7.5. Other tasks were completed without any obstacles.

7. TESTING

tasks/students	student 1	student 2	student 3	student 4	student 5
task 1	✓	✓	✓	✓	✓
task 2	✓	✓	✓	✓	✓
task 3	✓	✓	✓	✓	✓
task 4	✓	✓	✓	✓	✓
task 5	✓	✓	✓	✓	✓
task 6	!	✓	!	!	!
task 7	✓	✓	✓	✓	✓
task 8	✓	✓	✓	✓	✓
task 9	✓	✓	✓	✓	✓
task 10	✓	✓	✓	✓	✓
task 11	✓	✓	✓	✓	✓
task 12	✓	✓	✓	✓	✓
task 13	✓	✓	✓	✓	✓
task 14	✓	✓	✓	✓	✓
task 15	✓	✓	✓	✓	✓
task 16	✓	✓	✓	✓	✓
task 17	✓	✓	✓	✓	✓
task 18	✓	✓	✓	✓	✓
task 19	✓	✓	✓	✓	✓

Table 7.4: Task completion of students

- ✓ Success
- ! Success with difficulty
- x Failure

7.3.4 Changes proposed by participants

During testing, some more active participants gave suggestions on how to improve the user interface. Their suggestions are documented in Table 7.5.

Table 7.5: Changes proposed by participants

Proposed change	Reason	Priority
Replace lockable drag-and-drop field of subjective proficiency with a three-dot icon suggesting a draggable field.	It is more intuitive and common for material design.	high
Add a placeholder to the skill name input field, in parameters for filtering, with text 'Type skill name'.	It is not clear that the element is an input field.	high

7.3. Usability testing

Proposed change	Reason	Priority
Change positioning of elements section of filtering by skills.	Current positioning of elements is odd and a little confusing, it hides the skill name input field.	high
Add a placeholder 'search by name' to the search bar.	It is not clear what is the expected input.	high
Highlight current page in menu.	Menu item is only slightly visible and it is not clear which page is currently visited.	medium
Add a 'trash can' icon to delete reference, next to 'edit' icon.	It is not clear that users can delete a reference.	medium
Change the style of endorsing skills and viewing endorsements. Big blue button with thumb appears like action for listing endorsements.	It is not clear which actions endorses the skills.	medium
'Apply filter' button should also be at the bottom of the screen.	Last action the user performs, should be the application of filter. Therefore, it is logical to put it last, not first.	low
Search should show results continuously with each change to the search parameters, not only after application of the filter.	It is a more common behaviour.	low
Rename the filtering from 'search parameters' to 'filter parameters'.	Consistence with the rest of the page.	low
Add the option to select more skills in the skill cloud at once and then confirm via 'confirm' button.	Quicker skill selection.	low
Instead of global skill cloud for everyone, show personalized cloud with skills that the teacher, who is browsing the system, teaches in his/her courses.	Teachers of hardware have no intention to search people with skills not related to hardware.	low
Visibility settings could have the visible/hidden options surrounding the toggle button.	It would be clearer what the user is switching between.	low

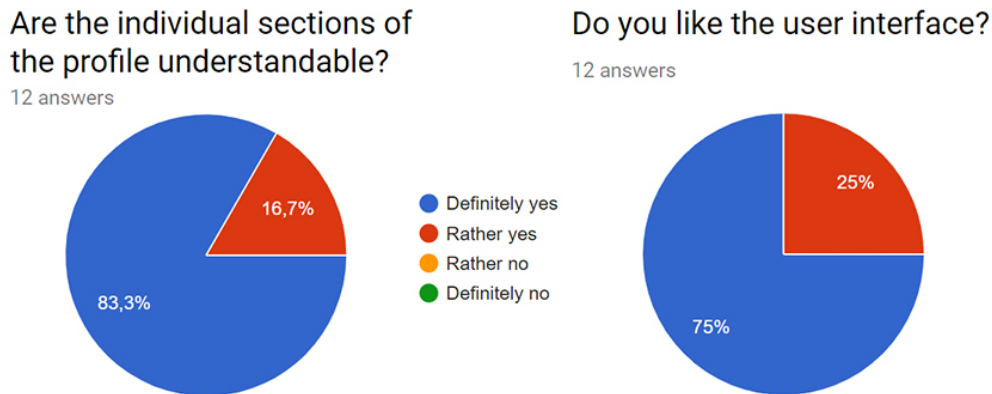


Figure 7.5: Graphs of participants' opinions about the user interface

Summary of the results

There were 12 test subjects in total: 4 company representatives, 3 teachers, and 5 students. The participants concluded that the user interface is pleasant and intuitive. This was partly due to the fact, that the design is in accordance with Material Design, which is commonly used in many modern applications, therefore, participants were seeing elements they are already familiar with. Although the testing uncovered a few problematic areas, the graphs, shown in Figure 7.5, illustrate that the overall satisfaction with the UI is very positive.

7.4 Acceptance testing

“Acceptance testing, in the engineering and software industries, is a functional trial performed on a product before it is put on the market or delivered, to decide whether the specifications or contract have been met.” [45] *“Usually, Black Box Testing method is used in Acceptance Testing. Testing does not normally follow a strict procedure and is not scripted but is rather ad-hoc.”* [46]

Testing was done on the latest version of the system which is shown in Figure 7.6 with an example of a current state of the profile page. More screenshots can be seen in Appendix B. The result was thoroughly tested with the supervisor to check if the agreed requirements were met, using the following checklist, which summarizes each functionality and its acceptance.

- UC1 – Skill based search of students

Description: Profiles contain generated skill sets. Users can choose a set of skills and system finds profiles with the highest match.

Result: Accepted

- UC2 – Profile setup
Description: Students can add, remove, and change order of their subjective proficiency.
Result: Accepted
- UC3 – Skill endorsement
Description: Teachers can endorse and also revoke endorsements of skills with an option to add a comment.
Result: Accepted
- UC4 – Manage reference
Description: Teachers can give references to students. They can also edit or remove given references.
Result: Accepted
- UC5 – Set profile visibility
Description: Students can set the visibility of their profile to each role. Visibility applies properly during a search, and hidden profiles do not appear in the results.
Result: Accepted
- UC6 – Search profile by name
Description: Users can search profiles by name. Autocomplete function is present.
Result: Accepted

Summary of the results

The solution was accepted at all points of testing. All goals, which were initially set for implementation, were met. The assignment is, therefore, considered as successfully fulfilled.

7. TESTING

Skills Search

Search Profile Settings Sign out

Name Surname

References

Ing. Name Surname 20.03.2019 02:04
"Skvělá spolupráce"

Ing. Name Surname 20.03.2019 11:34
"Nadprůměrná bakalářská práce. Student docházel na schůzky vždy včas a těžké problémy řešil samostatně."

Subjective proficiency

Specialization

- ✓ elasticsearch 1 👍

Main skills

- ✓ java 1 👍
- ✓ cassandra 1 👍

Other skills

- 👍 tests
- 👍 sql database
- ✓ modelling 1 👍

Generated skills

Graph Table

Developed by Adam Jankovec as a part of bachelor's thesis.

Figure 7.6: Profile page

Ideas for extensions

This chapter discusses possible ways of extending and improving the system.

Search engine

A core feature of the system, the search engine, should receive a major upgrade. An improved solution would allow filtering based on all of the parameters, listed in the Design chapter. The ordering of the results by the highest match should also favour those with more references, relevant endorsements, etc. The engine should be easily extendable with a new set of parameters. A teacher from BI-VWM⁶, who is knowledgeable of the field, recommended using the Spearman's rank correlation coefficient as a good starting point. Ing. Stanislav Kuznetsov proposed Elasticsearch [47] as another possible solution.

Skills structure

Another improvement could be creating a system, which would hold all the skills and support their management and versioning. Skills could be connected to other skills, with multiple types of connections. Such connections could represent skill similarity or hierarchy (superiority and subordination). Having a skill hierarchy would allow determining how knowledgeable a person is in some branch of industry.

Possible use case could be a user searching for a programmer, who is experienced with Kotlin⁷. If the system does not contain anyone who would match the request, the system could deduce Java as a skill similar to Kotlin and recommend Java programmers as candidates with suitable foundation for learning the requested skill.

⁶Searching Web and Multimedia Databases is a course taught at FIT CTU.

⁷A programming language.

Mapping jobs on skills

Support for mapping of courses on skills already exists in Grades. Mapping jobs on skills would make the task of searching for skilled people much easier. Instead of having to list all the required skills one by one, it could be possible to simply select “Java developer” as the only search parameter and get a list of people with a skill set similar to the one defined for such role.

This mapping would be a cornerstone to a much bigger feature. Students, during the event of deciding what courses to enroll, could select a position they would like to be prepared for. Selecting “Senior Java developer” as a dream position would first get a set of skills required for such position and then recommend courses, which provide the required knowledge.

Conclusion

The main aim of this thesis was to create a new light-weight system for a skill based search of students, while going through all stages of software development with respect to the methods of Software Engineering. Among the goals were: generating skill sets of students from educational data; allowing students to set their subjectively defined skill sets and letting teachers endorse these skills; allowing teachers to give references to students.

All thesis goals were met. Requirements of the target users were collected and documented. The system was successfully designed, implemented and tested and contains regression tests to mitigate the chances of introducing bugs in future development. To ensure safe manipulation with sensitive data, the system is secured with the industry standard protocol OAuth 2.0 using the authorization server of FIT CTU.

The user interface was designed according to a set of guidelines developed by Google, and successfully tested with participants from all the interested parties. Participants' suggestions were documented and will be used for the improvement of the UI.

The result is a new proof of concept, which consists of a responsive client (written with Angular), which communicates via REST API with a secured server (written in Java). The system is already integrated with existing systems running at FIT CTU, and the implementation was done using technologies in their latest versions, guaranteeing long-term support.

Implementing the whole system would exceed the scope of bachelor thesis, so further development will be undertaken by the Grades' development team, to make the system ready for deployment. Skill-based search of students will be used for recommending relevant students to assignments from both the teachers and the companies. The result could lead to an increase in work experience of students and also has a potential for attracting new faculty partners and possibly change the way faculties and companies cooperate.

Bibliography

- [1] International Educational Data Mining Society. *Educational Data Mining*. [Online]. 2011. URL: <http://educationaldatamining.org/> (visited on 04/04/2019).
- [2] Kuznetsov, S. *Ontologies and Recommender Systems in Educational Data Mining*. Doctoral study report. Prague: Faculty of Information Technology, Czech Technical University in Prague. 2015. (visited on 04/01/2019).
- [3] Faculty of Information Technology, Czech Technical University in Prague. *Klasifikace*. [Software]. 2018. URL: <https://grades.fit.cvut.cz>.
- [4] Data preprocessing, text mining and Ontology generating process. Kuznetsov, S. *Ontologies and Recommender Systems in Educational Data Mining*. [Figure]. Doctoral study report. Prague: Faculty of Information Technology, Czech Technical University in Prague. 2015. (visited on 04/01/2019). [Path: List of figures, Figure 3-1].
- [5] Balák, Z. *Portál pro podporu studia a klasifikace studentů*. Master thesis. Prague: Faculty of Information Technology, Czech Technical University in Prague. 2017. (visited on 04/01/2019).
- [6] Faculty of Information Technology, Czech Technical University in Prague. *ProgTest*. [Software]. URL: <https://progtest.fit.cvut.cz/>.
- [7] Microsoft. *LinkedIn*. [Software]. May 2003. URL: <https://www.linkedin.com> (visited on 04/05/2019).
- [8] LinkedIn Corporation. *About LinkedIn*. [Online]. ©2010. URL: <https://about.linkedin.com> (visited on 04/02/2019).
- [9] Stack Overflow. *Candidate Search*. [Software]. ©2017. URL: <https://www.stackoverflowbusiness.com/talent/platform/source/candidate-search> (visited on 04/05/2019).

- [10] Stack Overflow. *About Stack Overflow*. [Online]. ©2017. URL: <https://www.stackoverflowbusiness.com/talent/about> (visited on 04/02/2019).
- [11] Faculty of Information Technology, Czech Technical University in Prague. *Spolupráce s Průmyslem*. [Software]. ©2014. URL: <https://is.fit.cvut.cz/group/ssp> (visited on 04/05/2019).
- [12] Liferay Inc. *Liferay Digital Experience Platform*. [Software]. ©2019. URL: <https://www.liferay.com/products/dxp> (visited on 04/05/2019).
- [13] Czech Technical University in Prague. *KOmpONENTA Student*. [Software]. URL: <https://kos.cvut.cz> (visited on 04/05/2019).
- [14] Oracle. *JDK 12*. [Online]. Mar. 2019. URL: <https://openjdk.java.net/projects/jdk/12/> (visited on 04/02/2019).
- [15] Colebourne, S. *Should you adopt Java 12 or stick on Java 11?* [Online]. Oct. 2018. URL: <https://blog.joda.org/2018/10/adopt-java-12-or-stick-on-11.html> (visited on 04/02/2019).
- [16] Java Community Process. *JSR 338: Java Persistence 2.2*. [Online]. Mar. 2019. URL: <https://jcp.org/en/jsr/detail?id=338> (visited on 04/02/2019).
- [17] Oracle. *Java EE 5 Tutorial*. [Online]. ©2010. URL: <https://docs.oracle.com/javase/5/tutorial/doc/bnbqa.html#bnbqb> (visited on 04/03/2019).
- [18] Red Hat Middleware, LLC. *Java EE 5 Tutorial*. [Online]. ©2004. URL: <https://docs.jboss.org/hibernate/orm/3.3/reference/en/html/queryhql.html> (visited on 04/03/2019).
- [19] Stack Overflow. *Developer Survey Results 2019*. [Online]. 2019. URL: <https://insights.stackoverflow.com/survey/2019> (visited on 04/03/2019).
- [20] Pivotal. *Spring Framework*. [Online]. Mar. 2019. URL: <https://spring.io/projects/spring-framework> (visited on 04/02/2019).
- [21] community. *Overview of Spring Framework*. [Online]. June 2017. URL: <https://docs.spring.io/spring/docs/5.0.0.RC2/spring-framework-reference/overview.html> (visited on 04/03/2019).
- [22] Rouse, M. *REST (REpresentational State Transfer)*. [Online]. Dec. 2017. URL: <https://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer> (visited on 04/02/2019).
- [23] Parecki, A. and collective. *OAuth 2.0*. [Online]. Feb. 2019. URL: <https://oauth.net/2/> (visited on 04/02/2019).
- [24] Parecki, A. and collective. *OAuth 2.0 Authorization Code Grant*. [Online]. Aug. 2018. URL: <https://oauth.net/2/grant-types/authorization-code/> (visited on 04/02/2019).

-
- [25] community. *Typescript*. [Online]. Mar. 2019. URL: <https://www.typescriptlang.org/> (visited on 04/02/2019).
- [26] ostezer and Drake, M. *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems*. [Online]. Mar. 2019. URL: <https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems> (visited on 04/02/2019).
- [27] Lucid Software Inc. *Database Structure and Design Tutorial*. [Online]. 2019. URL: <https://www.lucidchart.com/pages/database-diagram/database-design> (visited on 04/02/2019).
- [28] Chapple, M. *Database Normalization Basics*. [Online]. Jan. 2019. URL: <https://www.lifewire.com/database-normalization-basics-1019735> (visited on 04/02/2019).
- [29] Valenta, M. *Normalizace a normální formy*. [Online]. Aug. 2018. URL: <https://courses.fit.cvut.cz/BI-DBS/materials/slides/pres-les08-normalizace.pdf> (visited on 04/02/2019). [File accessible after signing in to the CTU network].
- [30] Richards, M. *Software Architecture Patterns*. O'Reilly Media, Inc., Feb. 2015. ISBN: 9781491971437. URL: <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/> (visited on 04/02/2019).
- [31] Angular Team at Google and community. *Architecture overview – Angular.io*. [Online]. ©2010-2019. URL: <https://angular.io/guide/architecture> (visited on 04/02/2019).
- [32] Google. *Material Design*. [Online]. URL: <https://material.io/develop/> (visited on 04/02/2019).
- [33] Interaction Design Foundation. *Material Design*. [Online]. URL: <https://www.interaction-design.org/literature/topics/material-design> (visited on 04/02/2019).
- [34] Totie and collective. *Material Design*. [Online]. Mar. 2018. URL: https://commons.wikimedia.org/wiki/File:Material_Design.svg (visited on 04/02/2019).
- [35] Richards, M. *Layered architecture pattern*. [Online]. Feb. 2015. URL: <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html#idp782544> (visited on 04/02/2019). [Path: Chapter 1, Figure 1-1].
- [36] The App Assembly. *Word Cloud Visualisations for Splunk*. [Online]. 2016. URL: <https://theappassembly.com/word-cloud/> (visited on 04/02/2019).

BIBLIOGRAPHY

- [37] Faculty of Information Technology, Czech Technical University in Prague. *Apps Manager*. [Software]. ©2014. URL: <https://auth.fit.cvut.cz/manager> (visited on 04/05/2019).
- [38] Strumpflohner, J. *Local Angular CLI dev server with active proxy*. [Online]. Nov. 2016. URL: <https://juristr.com/blog/2016/11/configure-proxy-api-angular-cli/> (visited on 04/02/2019).
- [39] DbUnit Team and contributors. *About DbUnit*. [Online]. Nov. 2018. URL: <http://dbunit.sourceforge.net/> (visited on 04/02/2019).
- [40] Bellairs, R. *What Is Static Code Analysis?* [Online]. June 2018. URL: <https://www.perforce.com/blog/qac/what-static-code-analysis> (visited on 04/02/2019).
- [41] Rouse, M. *static code analysis*. [Online]. Nov. 2006. URL: <https://searchwindevelopment.techtarget.com/definition/static-analysis> (visited on 04/02/2019).
- [42] community. *About SonarQube*. [Online]. ©2008-2019. URL: <https://www.sonarqube.org/about/> (visited on 04/02/2019).
- [43] community. *SonarQube*. [Online]. ©2008-2019. URL: <https://www.sonarqube.org/> (visited on 04/02/2019).
- [44] U.S. Dept. of Health and Human Services. *Usability Testing*. [Online]. 2006. URL: <https://www.usability.gov/how-to-and-tools/methods/usability-testing.html> (visited on 04/02/2019).
- [45] Kenton, W. *Acceptance Testing*. [Online]. Apr. 2018. URL: <https://www.investopedia.com/terms/a/acceptance-testing.asp> (visited on 04/04/2019).
- [46] Software Testing Fundamentals. *Acceptance Testing*. [Online]. Copyleft 2019. URL: <http://softwaretestingfundamentals.com/acceptance-testing/> (visited on 04/04/2019).
- [47] Elasticsearch B.V. *Elasticsearch*. [Software]. ©2019. URL: <https://www.elastic.co/products/elasticsearch> (visited on 04/05/2019).

Acronyms

API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
CRUD	Create Read Update Delete
CTU	Czech Technical University in Prague
DAO	Data Access Object
DTO	Data Transfer Object
DW	Data Warehouse
DXP	Digital Experience Platform
FIT	Faculty of Information Technology
FK	Foreign Key
HTTP	Hypertext Transfer Protocol
HQL	Hibernate Query Language
IDE	Integrated Development Environment
IT	Information Technology
JSON	JavaScript Object Notation
KOS	KOMponenta Student
LTS	Long Term Support
NF	Normal Form
ORM	Object-Relational Mapping

A. ACRONYMS

PK Primary Key

REST REpresentational State Transfer

SSP Spolupráce S Průmyslem

SQL Structured Query Language

UI User Interface

URL Uniform Resource Locator

URI Uniform Resource Identifier

XML Extensible Markup Language

Screenshots of the result

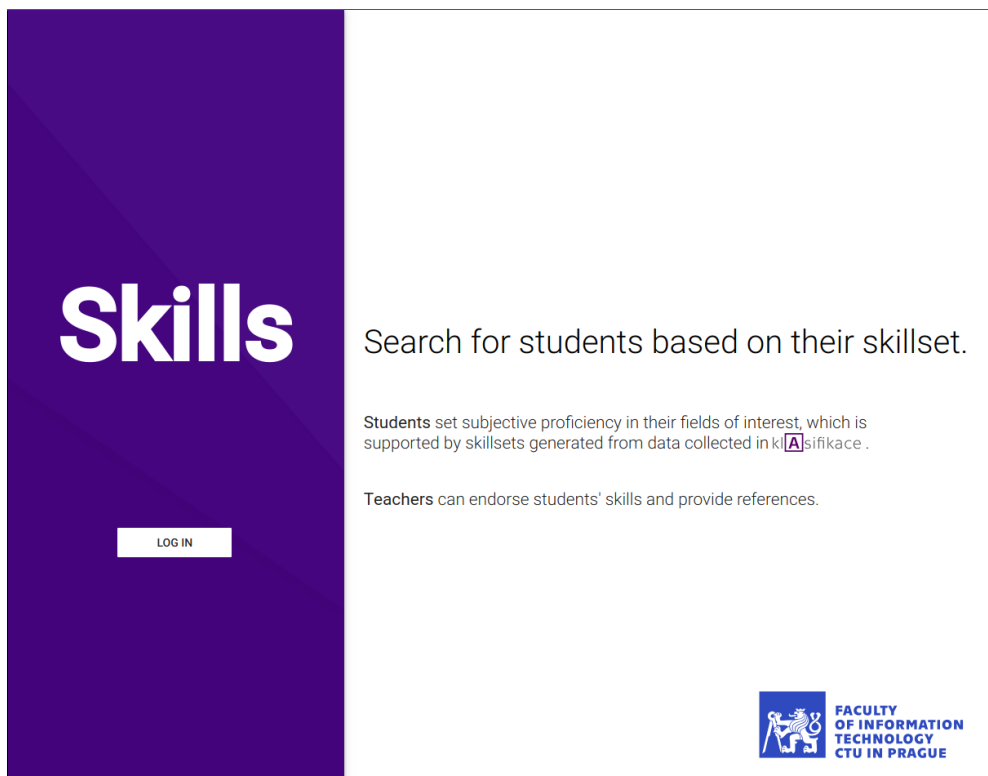
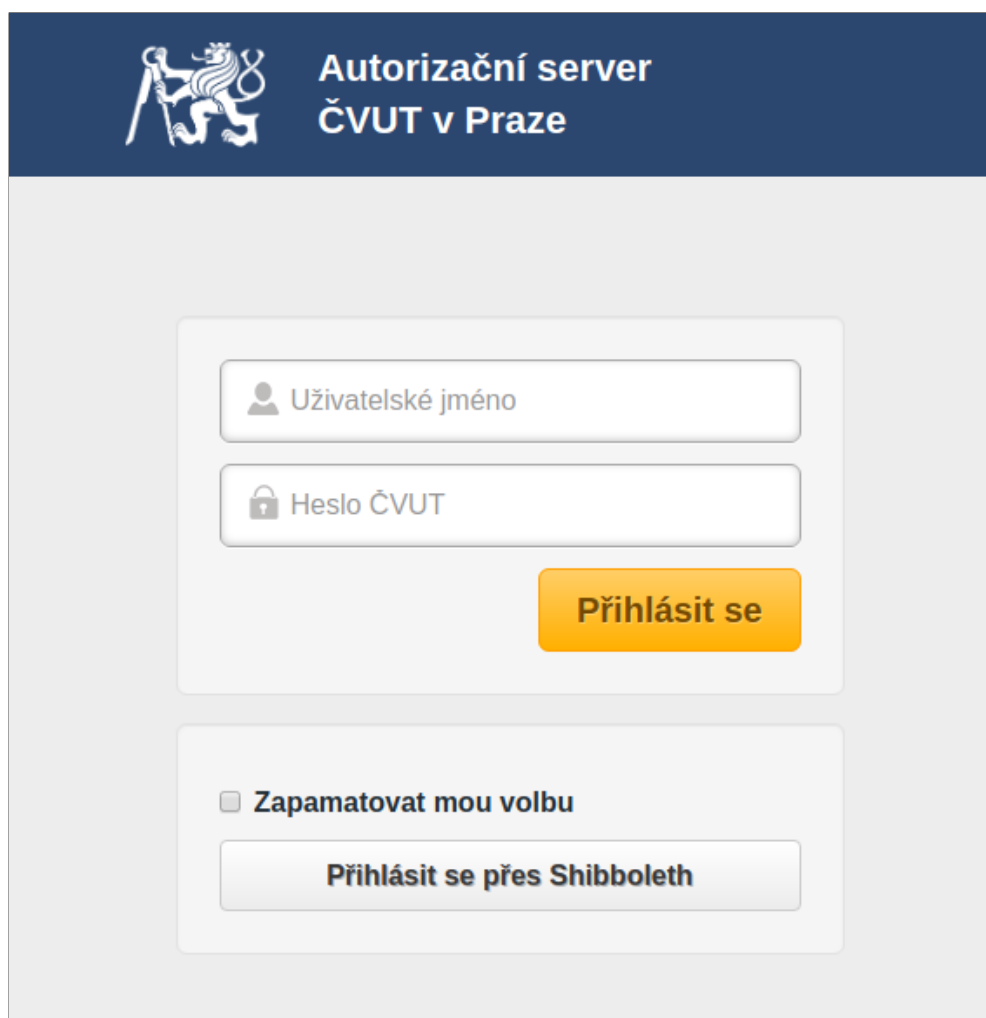



Figure B.1: Login page



 **Autorizační server
ČVUT v Praze**

Přihlásit se

Zapamatovat mou volbu

Přihlásit se přes Shibboleth

Figure B.2: Authorization page

☰ Skills
🔍 Search

- 🔍 Search
- 👤 Profile
- ⚙️ Settings
- 🚪 Sign out

Name Surname

References +

- Ing. Name Surname**
"Skvělá spolupráce"

20.03.2019 02:04
- Ing. Name Surname**
"Nadprůměrná bakalářská práce. Student docházel na schůzky vždy včas a těžké problémy řešil samostatně."

20.03.2019 11:34

Subjective proficiency

Specialization

- ✓ elasticsearch
1 👍

Main skills

- ✓ java
1 👍
- ✓ cassandra
1 👍

Other skills

- tests
- sql database
- ✓ modelling
1 👍

Generated skills

Graph
Table

Developed by Adam Jankovec as a part of bachelor's thesis.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE


Figure B.3: Profile page

Edit subjective proficiency

Add a new skill

Skill name

Manage skills

Rows are locked 

Specialization

java

javafx

Main skills

Other skills

ba

base

ga

tests

Figure B.4: Edit subjective proficiency page

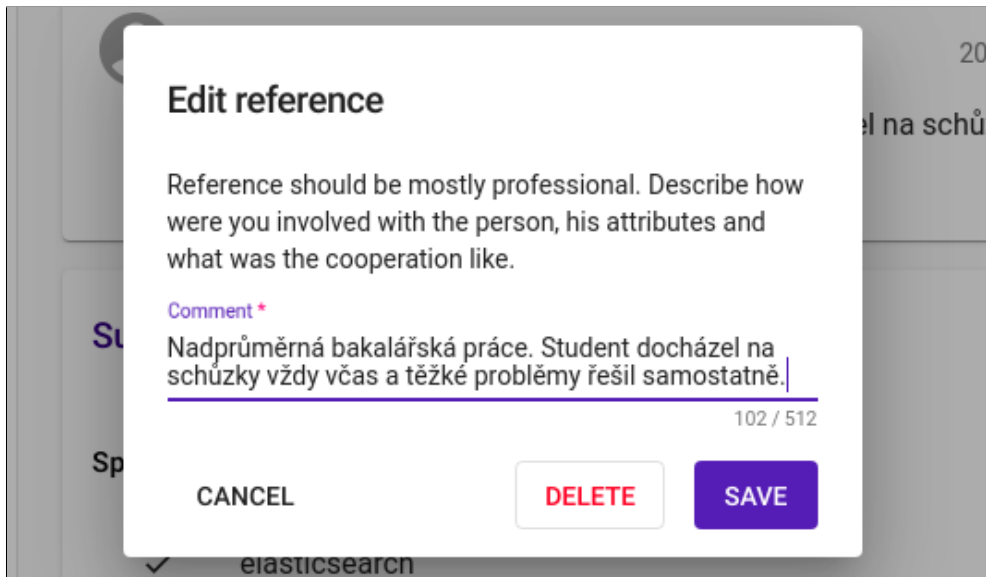


Figure B.5: Edit reference dialog detail

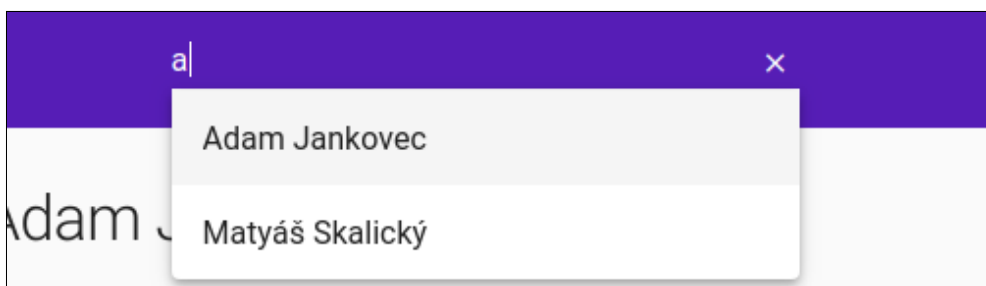


Figure B.6: Search bar with autocomplete detail

B. SCREENSHOTS OF THE RESULT

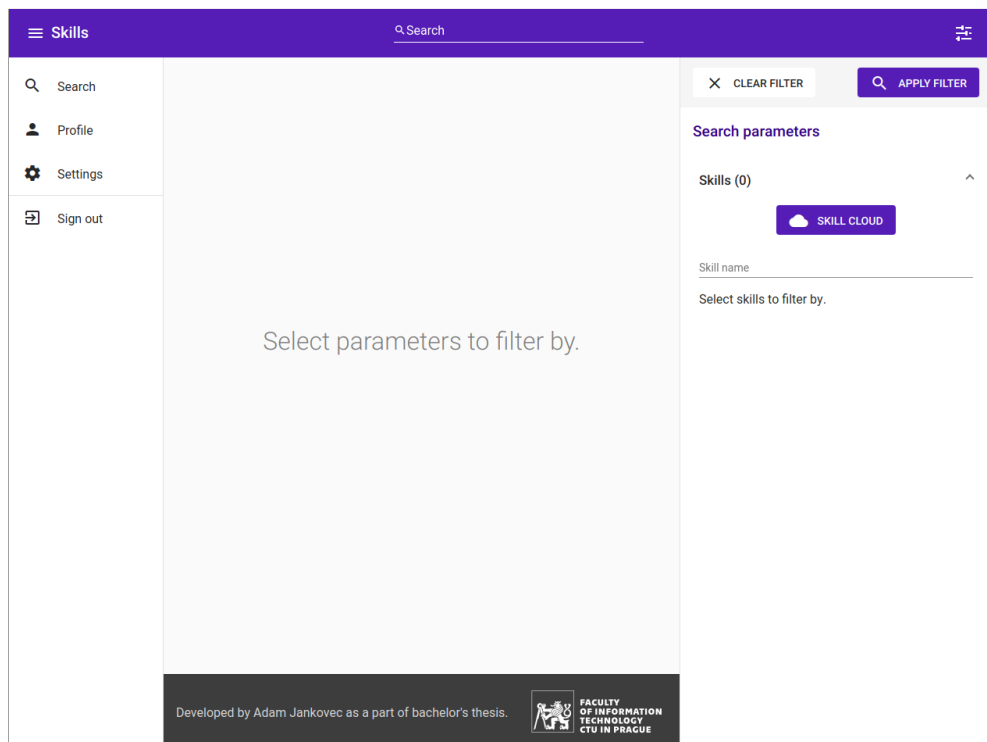


Figure B.7: Search page without parameters

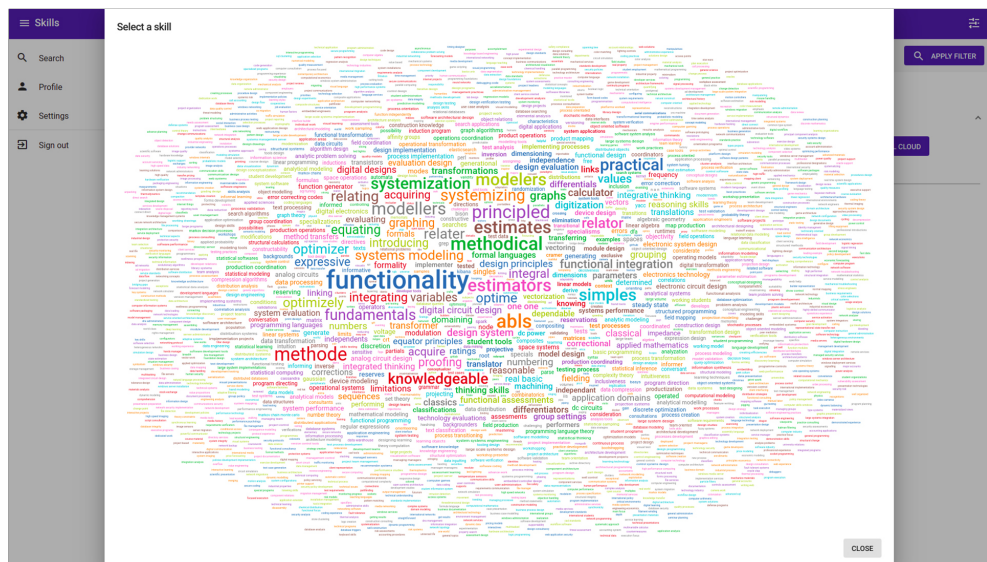


Figure B.8: Skill cloud for browsing most used skills

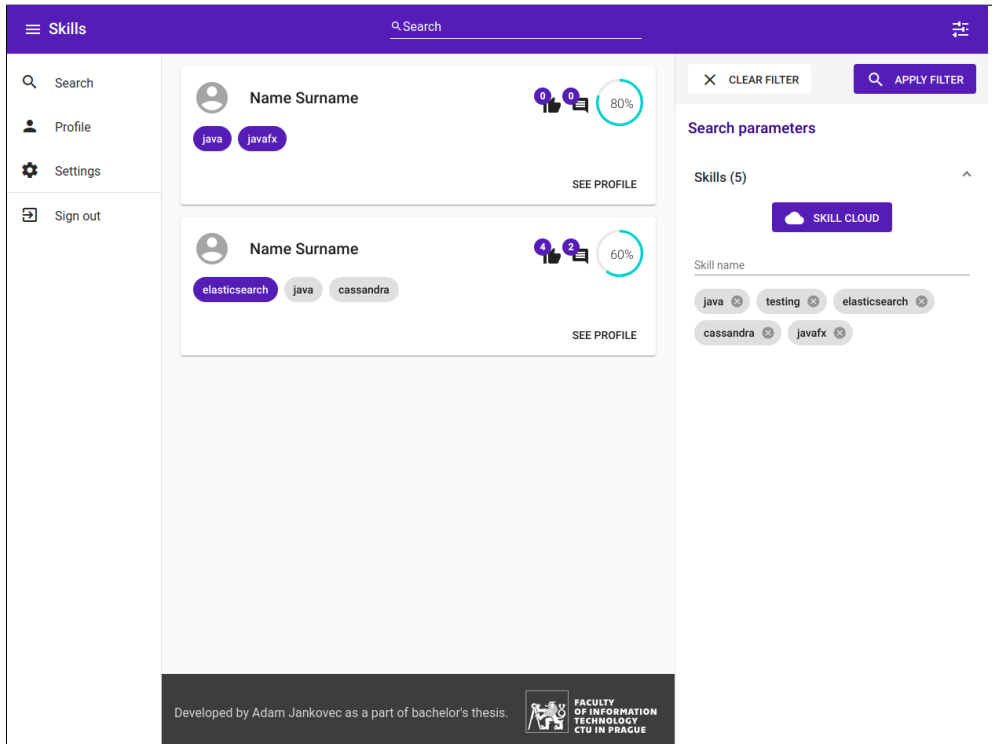


Figure B.9: Search page with parameters and results

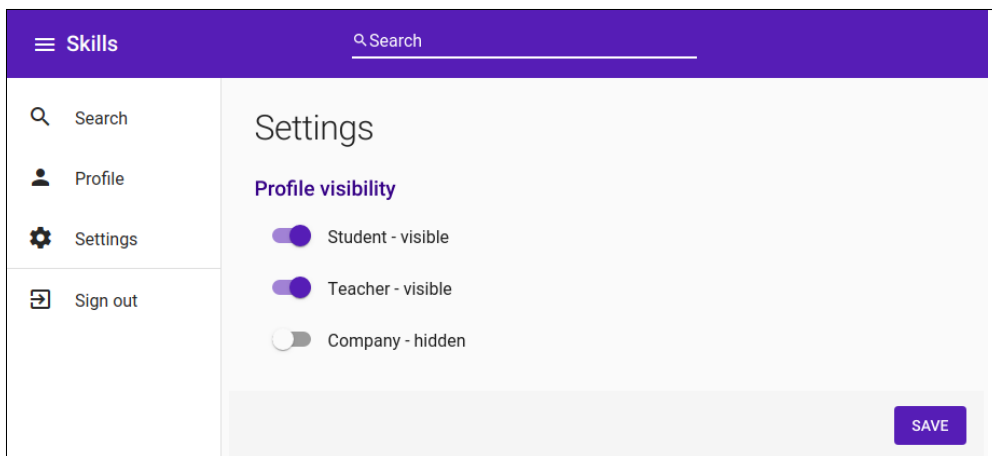


Figure B.10: Settings page

REST API definitions

Endorsement controller

users/{username}/skills/{skill-id}/endorsements

- Method – PUT
- Description – Endorse user's skills

users/{username}/skills/{skill-id}/endorsements

- Method – GET
- Description – Get endorsements of user's skills

endorsements/{endorsement-id}

- Method – DELETE
- Description – Delete endorsement

User controller

users/{username}/info

- Method – GET
- Description – Get information about specific user

user-info

- Method – GET
- Description – Get information about logged in user

user-roles

- Method – GET
- Description – Get roles of logged in user

users/{name}/autocomplete

- Method – GET
- Description – Get list of users, whose names match the query

Reference controller

users/{username}/references

- Method – POST
- Description – Add reference to the user

users/{username}/references

- Method – GET
- Description – Get user's references

references/{reference-id}

- Method – PUT
- Description – Update reference

Search controller

search

- Method – GET
- Description – Get profiles matching the parameters

Settings controller

settings

- Method – GET
- Description – Get settings of logged in user

settings

- Method – PUT
- Description – Update settings of logged in user

Skill controller

skills/{name}/autocomplete

- Method – PUT
- Description – Get list of skills, whose names match the query

SkillPoints controller

users/{username}/skill-points

- Method – GET
- Description – Get user's generated skills with points

skill-points

- Method – GET
- Description – Get global skills with points

SubjectiveProficiency controller

users/{username}/proficiency

- Method – GET
- Description – Get user's subjective proficiency

skill-points

- Method – PUT
- Description – Set user's subjective proficiency

Contents of enclosed DVD drive

	readme.txt	the file with DVD drive contents description
	exe	the directory with executables
	src	the directory of source codes
	impl	implementation source codes
	thesis	the directory of \LaTeX source codes of the thesis
	attachments	the directory of attachments for project setup
	text	the thesis text directory
	thesis.pdf	the thesis text in PDF format